

SINCLAIR



WORLD

SEPTEMBER 1992 £1.95

POOLING RESOURCES

Putting the odds in your favour with the QL

QL Users Dictionary

VOCAB

SuperBasic interpreter vocabulary reports - AUTOMATICALLY!

QDesign

Advanced Graphics and Design for your QL

QL World (*kew-el wirld*): The definitive guide to the world of the QL Computer. With software and hardware reviews, book reviews and news from the world of Sinclair computing.

Toolkit (*dee -eye -why tulekit*): Every QL user needs a good guide to building useful QL functions. The September issue covers VOCAB.

VOCAB (*vocab*): An essential program, automatically generating formatted reports on the vocabulary of your SuperBasic interpreter.

QL Scene (*kew-el seen*) C68 source code now public... translators wanted... Sir Clive's bike briefly baulked... optical character non-recognition.

bleshooter (*trub-el shu-ter*): Archive modification, big dictionaries and yet more disc situations. And where have all the grumbles gone? It must be the holiday season.

Open Channel (*oh-pen chan-el*): Positive feedback from our readers.

Pooling Resources (*pu-ling re-sor-ses*) The first thing Stan Canton did when he bought his QL was build it a home of it's own. Since then it has been assisting him with his business and software development, including Football Pools Forecas

QDESIGN (*kew deesine*): QDesign advanced graphics and design pro Jochan Merz, which uses the QJu environment. A. F. Wilson has QDesign 1 for about a year.

One man's system (*juan ma*)... uses Abacus

QDESIGN

advance from Jo pointer using C

One David Dublin tracking accou

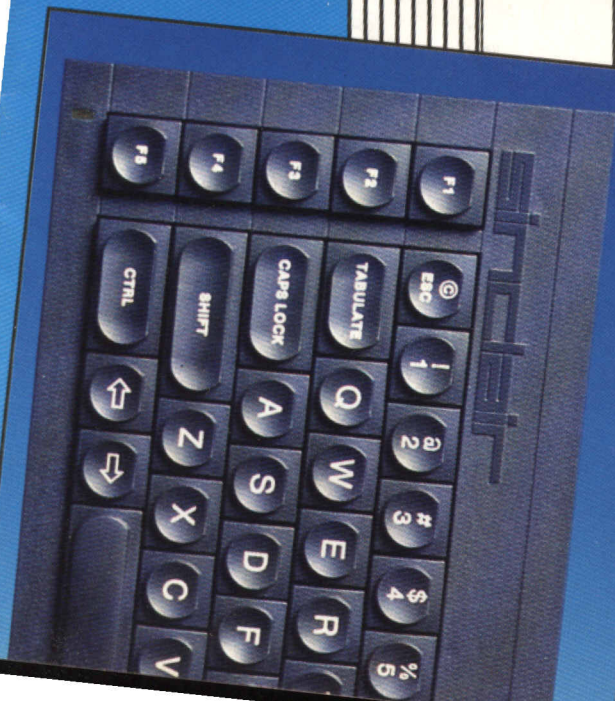
A Qu Begin how deter repre

DIY more fun VO Vo pro use

Q gu sc re c I r



9



Editor
Helen Armstrong

Publisher
Mark Kasprovicz

Advertising Manager
Jim Peskett

Creative Director
John Stanley

Designer
Steve Billington

Magazine Services
Pauline Wakeling
Frances Maxwell
Linda Miller

Sinclair QL World,
Published By Arcwind Ltd.
The Blue Barn,
Wootton, Woodstock,
Oxon. OX7 1HA
Tel: 0993 811181
Fax: 0993 811481
ISSN 026806X

If you have any comments or difficulties please write to the Editor and we will do our best to deal with your problem in the magazine, though we cannot guarantee individual replies.
Back issues are available from the publisher price £2.50 UK, £2.99 Europe. Overseas rates on request.

Subscriptions from: Arcwind
The Blue Barn, Wootton, Woodstock,
Oxon OX7 1HA
UK: £23.40
Europe: £32.90
Rest of World: £40.90

Reprographic Services: The Wace Group, London.
Printed and Bound by
William Gibbons, Planetary Road, Wil-
lenhall, West Midlands
Distributed by: IPC Market force,
King's Reach Tower, Stamford St,
London SE1 9LS.

© 1992 Arcwind Ltd.
Sinclair QL world is published 12 times a year. All rights reserved. Reproduction in whole or in part without written permission is strictly prohibited. We welcome contributions but can not be responsible for any unsolicited contributions, all material must be supplied with a SAE.

CONTENTS

■ ■ SEPTEMBER 1992

4 TROUBLESHOOTER

Archive modification, big dictionaries and yet more disk situations. And where have all the grumbles gone? It must be the holiday season.

11 QL SCENE

C68 source code now public ... translators wanted ... Sir Clive's bike briefly baulked ... optical character non-recognition

12 POOLING RESOURCES

The first thing Stan Canton did when he bought his QL was to build it a home of its own. Since then it has been assisting him with his businesses and software development, including a Football Pools Forecast program.

13 SOFTWARE FILE - COWO QTOP

Dell Allan runs through the latest update on Cowo's QTop front end for QL and Thor computers.

14 OPEN CHANNEL

16 ONLY FOLLOWING ORDERS

Following the first part in July QL World, P Warne shows how Abacus can keep track of bank payments that do not occur regularly every month.

22 USER REPORT - QDESIGN

QDesign is an advanced graphics and design program from Jochen Merz, which utilises the QJump Pointer Environment. A F Wilson has been using QDesign 1 for about a year.

26 THE NEW USER GUIDE - part 18

Section Eighteen comes with a surprise - the contents of Section Sixteen. These should have been in the July issue of QL World. Section Eighteen appeared in Section Seventeen last month. We hope you understand.

33 DIY TOOLKIT

Simon Goodwin's VOCAB generates formatted reports on the vocabulary of your SuperBasic interpreter.

37 ONE MAN'S SYSTEM

David McCullagh uses Abacus to keep his Dublin swimming team afloat. As well as tracking the swimmers, it keeps track of the accounts as well.

40 ARCHIVE ANSWERS - BIBLIOGRAPHY

Robin Stevenson's method for using Archive to structure a set of bibliographical references can also be applied to other collections.

43 SYSTEMATIC MACHINE CODE PROGRAMMING Part 10

COMING SOON!

After all that stuff with windows - VOCAL is the best one, followed not very closely by One Man's System, and Stan.

TROUBLE

Bryan Davies gets a B in his bonnet. The editor says: "Oh, K."

Hardly had the August *Trouble shooter* gone off to the magazine than my comments about not using *Archive* started to seem less funny. The time had come to try to do something with a set of accounting routines written in the *Archive* language, years ago, and it was soon clear that this was no easy task. Not to put too fine a point on it, I gave up, but only after several days of struggle. Maybe the original programmer wanted to keep the code obscure in order to protect his commercial interest, but that probably wasn't the main reason for my difficulty in following the thread.

We all have our trouble trying to think straight. We usually find it easier to get stuck straight in to the job, as we see first see it. Only later do we realise the agonies created by not having a plan or flow diagram. You can get away with unstructured programming when the program to be written is short and simple, but this program ran to over 70 pages of procedures. It wasn't really the program as such that caused the problems; it took so long sorting out relatively minor errors and making the screen presentation acceptable that there was little time left to consider the operation.

That may sound like a lack of planning, but maybe it was more due to my assuming the program worked properly, rather than being my normal sceptical self. The outcome was a partially-working program, better in its user interface but still with some gaping holes in it. Maybe some of them will be sorted out later, but I had to say that the contrast between working with *Archive* and with the program I use often on a N Other machine was painful.

The area which highlights the big weakness of *Archive* is that of testing Procedures and Screens. The language itself is good, but what a performance to modify and check what you write! Bear in mind that the program had several hundred small Procedures, combined into a couple of dozen groups. To modify anything in a group, you have to run *Archdev* (not *Archive*, since this program was to run under the run-time version, *Archrim*), Load Object "flp1_procname", scan through maybe a dozen individual Procedures, find the area that seems to need changing, make

the change, Save Object "flp1_procname", then Run Object "flp1_boot" to run the whole program, select the option concerned and see if it now works as desired, then go back through the whole process when it becomes clear something is still wrong.

Should you screw it so badly that there is no way out - easy enough with routines where the user has to press a key - you may have to reset, possibly making a hash of connected data files and having to start again from scratch. After doing this a few dozen times, with a few dozen groups, each with a dozen or so individual Procedures, one can get a bit browned off. In some ways, even so, this is easier than dealing with Screens, where the routine is similar but harder to get to grips with.

From the user's standpoint, such a program will probably work better in SuperBasic and compiled with DP's *Turbo*. The programmer will have security, too, as few users will be able to modify compiled code. However, you may be like me and feel able to write simple *Archive* lines but uncomfortable writing in SuperBasic. If that's the case, try observing a few "rules":

Draft a plan of what you have to achieve, and the routes to be taken to get to the various points on the way.

Create Procedure names that have as much meaning as possible - don't call a Procedure A2, for instance, when it is concerned with a listing name. Try something like LISTNAME_PRO.

Do likewise with Variables - don't use AQDV\$ when ORDERNO\$ is more to the point.

If you have sufficient ram to hold one large Procedure group, create just the one, and do away with duplication of common Procedures (eg for printing to the screen) in several groups.

Load both Procedures and data to ramdisk(s), to improve operating speed. You can use SuperBasic to WCOPY program and data files to ram then start the program from ram.

Write backup Procedures to copy the data file(s) to two floppy disks at intervals during working sessions - don't wait to the end of a session to do backups, as the end may come before you intend it to.

Use *Archdev* and *Archrim*, rather

than *Archive*, because *Archrim* takes less space and runs faster.

Archive on the QL will never be a fast operator. Try to get the best out of it by using the run-time version, providing it with as much ram as possible, and using the fastest QL you can get - both those point imply a Gold Card, of course.

Checking spelling

Perhaps some of my past comments on this operation gave a false impression of my feelings on dictionary size. My intention was definitely not to imply that biggest is best where word-list ("dictionary") files are concerned - far from it. Ken Whyld (*Open Channel*, July 1992) points to one dilemma, and the editor gave one solution for it. The larger the word list, the more "words" it is likely to contain that can mask incorrect spelling in documents. This is inevitable, as computer word lists are generated from dictionaries which contain common abbreviations, such as the one Whyld cites. "dod" - if written in capitals - could be short for Department Of Defense (US spelling, of course).

(My *Chambers* gives "dod" as "a slight fit of ill-humour; the sulks (often pl.) (Scots.); as "to poll, pollard or clip (obs or dial)"; and "a minced form of God, in Oaths" - all reasonable reactions to a spellchecker that won't pick up a mistyped "did" - Editor.)

It is unfortunate that many spellcheckers are deficient in their ability to handle capital letters, and this is a functional reason for the problem he is worried about; if the checking mechanism were improved, it would not matter so much that such words are in large lists. As both *Perfection* and *text87* come with large and small word lists, the user can decide what his/her main problem is - spelling errors or typos - and choose the size accordingly.

The weakness in handling capital letters regularly annoys me, but it may not trouble many users. It is primarily technical. For example, how do you write the abbreviations for megabytes, kilobytes and megahertz? The latter is perhaps the easiest; there are international standards for electrical terms, accepted by engineers and scientists. Units of measure named

SHOOTER

after whoever created them are often allocated a capital, such as Hz for Hertz. Where the same initial-letter could be used for several different quantities, the larger - such as M for mega (millions) might have the capital and the smaller (Greek mu μ for millions) a lower-case or related symbol. A lower-case m denotes milli- (thousandths) as in millimetre. ("Thousands", on the other hand, uses a lower-case k, from the Greek "kilo" (km for kilometre).)

Where computers are concerned, the computer press doesn't seem too well versed in the efforts made by the engineering and scientific communities. You can see kilobytes shown as kb, KB, Kb or kB, and megabytes as mb, MB, Mb or MG. The latter is surprising; presumably the "G" comes from mega, but the usage has to be wrong, surely? Another reason for capitals is the German manner of writing the full words, such as Kilohertz. As "byte" is not a name, a small letter seems appropriate, but I'm loath to change from my own normal usage of "B". To confuse the issue further, computer people have chosen - not without reason - to use "k" for a value which is not 1,000. A kilobyte is 1,024 bytes, and a megabyte is 1,024 by 1,024 bytes (2 to the power of 20).

My own electro-technical German-English dictionary appears to accept both capitals and small letters for the qualifiers, so maybe there is not as yet a "right" way of putting them.

(The Editor comments: I'm puzzled by some of this. The computing and engineering journals I have worked with over the years have been careful with standard symbols. Engineers (electronics and computing) and professional journalists - in the UK and USA, at any rate - have a separate standard symbol for binary thousand (1,024): a capital K. The same symbol also denotes degrees Kelvin, a measure of temperature, which is not easily confused with binary thousands.

The lack of an accepted standard for "bits" and "bytes" is a pest. We use MB for megabytes, and Mbits or Mb for megabits. I've never seen MG for megabytes. German capitalises many common nouns, but that shouldn't affect the use of symbolic letters.

Many compilers of technical standards think that engineers have an attitude too narrowly focussed on their own local practices or areas of expertise. Many engineers think that writers of technical standards are wazzocks; I agree with both of them, but as most engineers and technical writers I personally work with handle analogue, digital, hardware and software development simultaneously, I pay close attention to their usages.)

... And now, back to

Returning to the spellchecking function, to cope with abbreviations, the checker needs to be able to recognise capital letters, which may be at the start or end of words, or in the middle. Some checkers can recognise that, for instance, "Germany" is correct and "germany" is incorrect, but that's about the limit, and this is where the large word list can let you down.

In practice, 10-20,000 words is enough, but a large separate list may need to be added if the user has a good collection of abbreviations and proper nouns (such as "Joseph", "Glasgow", "Safeways"). My own experience has been that no more than about 10,000 words are added to a bought word list, and the same number are usually added to any new list - regardless of size - simply because they are almost all abbreviations or proper nouns which are rarely included in commercial products.

The larger of the various *Perfection* word lists are an exception, as they have had many such words added to them and will generally not be caught out by common words such as "Sinclair" or "QL"! There is some merit, then, in having bigger lists, and this is especially true if you work in a trade which has its own vocabulary, or you make use of less common words, for whatever purpose.

Don't assume, though, that all strange words in your own particular field will be included in a list which has over 100,000 words. In practice, you can find many quite normal words missing from 150,000-word lists, and even 300,000-plus lists will not contain all words used in specialist areas.

Cross-eyed OCR

OCR (optical character recognition) has been a Holy Grail in the electronics development world for many years. Younger computer freaks may not realise how long such facilities take to develop. In my own experience, OCR ranks with blind landing as something which has taken a surprisingly long time to come to commercial fruition. If one ignores pioneer experiments, blind landing systems development dates from the early 1940s. We were working very seriously with two commercial system philosophies in the UK in the early 1960s, but such systems have reached the level of common, reliable use only in the past few years. Attempts at OCR presumably date back at least as far; certainly, I worked on production reading machines at the end of the 1960s.

This doesn't alter the fact that reading technology still has far to go, and you need to ignore much of the verbiage on the subject. There are several OCR components on sale, and a very few capable software products to go with them, but they all suffer from basically the same constraints as before. The printed or written text needs to be of good quality and consistent form. What brought this to mind was the editor's explanation of the gremlins that got into an article in the May/June issue.

It may be fairly easy to scan text into a computer in the form of one image, but that's a long way from translating that image into individual characters and feeding them in the correct format into a word-processing program. Performance figures of over 90% accuracy are commonly quoted for quite cheap OCR readers, and that sounds good at first reading. When you calculate how many errors that means per thousand words of text, it looks much less good. An article of the size of this one would have around 2,000 individual characters wrong at a reading accuracy of 90%. In the worst case, that would mean nearly 70% of the words having an error. A good typist can manage over 100 words per minute and will have been graded to an accuracy of 98%. At least, with OCR, errors may do no more than cause laughter, whereas even a small error when an airliner is landing in zero visibility can mean disaster.

Disks again

R Snow reported his problem with 5.25-inch floppy disks, in the May-June issue, and his conclusion - that the trouble was with the drive not the disks - seems the correct one. Being very much of the "watch the pennies" inclination myself, I never consign a floppy to the bin without a big fight, but one has now defeated me - the first in more than six years (and close to 1,000 disks). It was an old Philips-brand 5.25-inch 360 KB one that has held the Turbo-DOS operating system in its youth. Out of curiosity, and for the benefit of clients who know little of the innards of disks, I opened it up and was mildly surprised to see what seems like a paper-thin piece of dishcloth between each side of the casing and the magnetic platter.

The platter itself is very flexible and it is not surprising to hear of disks being doubled over and still coming up smiling when used afterwards - but don't try it. An example of this occurred about the same time, when a well-known (PC) software company sent me an upgrade to its operating system, in a flimsy A4 envelope, the 5.25-inch disks arriving with marked kinks in them but giving only a slight hint of trouble when used. In the case of my sole dud disk, the reason was clearly evident - there was scoring on one side, where the read/write head had apparently come into more-forcible contact than usual with the magnetic surface.

Given reasonable care, floppies really shouldn't give you trouble, even over periods of five years or more. During a recent re-organisation, some of my old 3.5 inch disks were reformatted, and files transferred between them, with no sign of problem; the files on them dated back as far as 1985 and had not been touched in the intervening years, other than being used read-only for tests, now and then. Surprisingly, one disk that had (not long ago) failed to format properly in one PC but had formatted correctly in the QL proved itself quite happy to format without error or murmur in another PC. Perhaps for the same reason, three or four commercial program disks which had behaved without problem in one computer gave reading errors in another.

Times change

What does seem to be true is that the quality of batches of disks varies appreciably, and having a known brand name on the casing is no guarantee of reliability. Some time ago, you might have been safe buying from the same supplier at intervals of months, even years, because your supplier might well have been buying from the same source all the time. This is unlikely to be the case now. For a few years, supply was greater than demand and dealers took

to buying job-lots from wherever they could, to keep their own prices competitive.

Obviously, quality tends to vary when the source is not constant. Recently, there have been regular reports of a general shortage of 3.5-inch disks, leading to some increase in prices, but this shouldn't be expected to improve the quality situation. The reason generally given for the shortage is that two software suppliers - IBM and Microsoft - have been selling so many copies of their operating system (OS/2) and front-end (Windows) programs, respectively, that they have taken a large slice of the disk supply. OS/2 version 2 comes on 21 high-density disks, no less! (Compare that with the economical two rom chips in which Qdos sits). The shortage doesn't seem to extend to DD disks, fortunately.

Those happy users having extra-high density (ED) drives may be able to get the disks at better prices before long, as the first non-IBM PC systems equipped with 2.88 MB drives have appeared in adverts. However, the (mainly-PC) computer magazine that claims to be "the best" still stated in a recent issue that ED disks cost £10 each - maybe PC users should buy from Miracle Systems. As Miracle provide 10 disks in their price of £175 (including VAT) for dual ED drives, that would seem to suggest they are almost giving the drives away.

One incidental, but significant, advantage of using ED drives is the space saved in storing disks. An ED disk holds 3.2 MB of files, which is more than four times what a DD disk holds; roughly, it takes only two ED disks to hold the same as nine DD ones. If you are short of workspace, this saving can make computing life much easier. One small box of ten ED disks from Miracle might well suffice to hold everything you need.

Demo disks

The new editor of *Quanta*, John Vanags, made what sounds like a good suggestion in his July issue. He made the obvious point that users are not keen to part with their money without knowing something about the goods first. Not all users can get to *Quanta* meetings and see software demonstrated by the suppliers. An alternative way of learning something about a program would be through a demonstration version of the program. Supplying demo versions of WP and other types of program has been common practice in the PC world for years, and it must certainly have influenced the buying decisions of a fair number of users. A demo version is essentially a cut-down version of the program, capable of showing major features but not usable for serious work by the recipient.

For example, you might be able to create documents, but not print them. One

very good demo disk I received allowed you to use many of the program features and even to print files which were supplied (but not ones you created). Other demos merely display program features without allowing the viewer to get involved, but this approach tends to make you go glassy-eyed quite quickly and seems much less effective as a selling tool.

By-and-large, the demo disks I have had were free; the only one I recollect paying for was from Microsoft (no surprise) and cost about £10. QL suppliers aren't in the same money league as PC ones, and demo disks would almost certainly have to be paid for, but the payment could be made refundable if the full version of the program concerned were purchased subsequently. Demo disks just might reduce the amount of piracy - it is very easy to ask a friend for a copy of a program, "to see what it is like". Any comments, suppliers?

Text87 manual

In response to various requests, Software87 have revised the instructions for the Plus-4 version of text87. There is more information on subjects which may have caused users problems previously, such as the Room command. A considerable number of changes have been made to the program during the transition from issue 1 through 2 to the current 3, and much of the added material concerns these changes. A welcome addition is a Glossary and Index. The on-screen Help information also has been extended. The program is, overall, quite a bit easier to use now than when it first came out; as indicated in last month's review, it is a very much more capable program than version 3 was and should find favour with both existing T87 users and late converts from *Quill*.

Name change?

Bill Richardson of EEC is now trading under his own name as well, according to his advert - from the same address as previously. Amongst other items of interest, he is offering hard disk drives, in either kit form or ready-boxed, and interfaces to allow various combinations of devices to be connected through the QL 64-way expansion port. He still has stock of QLs, and of both DD and HD 3.5-inch disk drives. The *QL World* readers' offer of new QLs at £69.95 is going like hot cakes.

Readers' letters

There has been something of a hiatus, with letters presumably going to the old addresses and taking a long time to catch up with us. Please check the new address for correspondence, at the front of the magazine. It is different from the one in the May-June issue.

PERFECTION SPECIAL EDITION

POWER

PERFECTION SPECIAL EDITION has 253 (two hundred and fifty three) direct/menu commands (not counting options in sub-menus), plus 32 special characters (like Bold on) that can be inserted 'directly' plus intelligent (and now excellently documented) macros. Comparisons with other word processors on the subject of power are hence quite unnecessary.

EASE OF USE

Independent reports, customer feedback and published reviews (of its less able but still excellent predecessor, PERFECTION) leave one in no doubt as to which word processor is friendliest — PERFECTION SPECIAL EDITION, with its intuitive, silky handling. Uniquely, it has two operating modes, with both menus (visible or invisible — they even look like Quill's) and direct commands (for when you familiarise yourself with the system). Uniquely, both modes are 're-entrant' (so you can use any menu option or direct command while you are in the middle of performing another option or command — block handling, etc, becomes a dream). Uniquely, PERFECTION SE has fully automatic memory management, grabbing and releasing RAM instantly as your document grows or shrinks — programs without this don't take full advantage of the multi-tasking abilities of the QL! Uniquely, PERFECTION SE leaves you in the driving seat, not juggling things around 'underfoot' while you are typing. Uniquely, PERFECTION SE allows up to nine different documents to be handled simultaneously from one copy of the program — each with totally independent margin, tab, justification, control panel, etc, settings. Uniquely, each document can itself have up to six environment settings, each settable or recallable instantly with a single keypress combination. Each document can have any number (up to 500,000 on GOLD CARD) of candidate blocks! Each document can have two independent windows (of any depth, of any (but same) width across) 'on to' it, even with overlapping text — that allows you to edit in one place while viewing another, to compare 'before editing' with 'after editing' (you can arrange to have one window remain 'frozen' in time), etc. Uniquely, we realise how much faster it is to type in something like CTRL/SHIFT/F5 than (say) F3 F3 R — both involve three keys, but as the former doesn't require the keys to be pressed in just one specific order, or to be released in any order at all (together will do), it is in practice twice as fast as the latter, where no key may be pressed until its predecessor is released. PERFECTION SE takes advantage of all this — it is the little things that count! Uniquely, by providing eight user-definable strips, PERFECTION SE allows you to cope with printers of the future, not just the printers that now exist — you can attach the strips to any printer features. Uniquely, PERFECTION SE's status lines give full information on all relevant global settings. And the manual has an index. Also, it has all the important bits at the front.

WYSIWYG?

By the latest definition of this term, neither is PERFECTION SE fully WYSIWYG, nor are other QL word processors. WYSIWYG means what you see on screen is exactly what you get on paper. Exactly — down to every wiggle in every character in every font.

To get true WYSIWYG, use PERFECTION SE's fully automatic link (supplied as part of PERFECTION SE) to PROFESSIONAL PUBLISHER, where you will get 100% WYSIWYG. 100%? Yes, 100%. With this combination, adjust the horizontal and vertical magnification on your monitor (ie calibrate it once and for all so screen circles correspond to same-diameter printed circles — poor monitors may distort a little bit at the edges). Now you can place your printed output from PERFECTION via PUBLISHER over your monitor screen, and get a match that is more perfect than is your eyesight. Now that is WYSIWYG.

SUPERB PRINT QUALITY & FLEXIBILITY

Uniquely, using the aforementioned automatic link, you can output PERFECTION SE documents using over a thousand fonts (a huge variety of styles and sizes, supplied on the PUBLISHER and TOOLBOX disks) on virtually any printer — from the humblest Epson RX80, Brother M1009 or Star LC10 (which are all single font machines when used with most word processors) to top-end lasers. *You are not limited to the fonts built into the printer!!* All PERFECTION SE **bold/underlined/italics/super/sub**, etc, settings are preserved. Proportional spacing and micro-justification are automatic, even when you mix fonts of differing widths and heights (even on the same line), vary line spacings, etc. Uniquely, you are not trapped with one type of micro-justification (ie adding all the space between words, and using the predefined widths of characters as their separation) — with our system, you can vary (in 5% steps) the proportion of micro-spaces added between words to that added between characters (the latter in proportion to their individual widths). Settings around 65%-35% — not the 100%-0% forced upon you by some other word processors — seem to give the most pleasing results. Uniquely, you are not limited to mere rectangular columns plus headers/footers — that's all the rest can do — you can output in any sequence to any number of frames (text flowing from one to the next), each of any shape — irregular polygons of up to 66 sides, circles, multi-column or part-column boxes (hundreds of types of borders, thousands of textures), doughnuts, wrap-around shapes, even re-entrant ones

(join-the-dots' type borders, even with intersecting edges) — all with micro-justification and proportional spacing! Look at the example on this page. Of course, if super fancy output or special effects are not of the essence, PERFECTION SE's direct printer output is more than capable of meeting your needs.

THE FASTEST

For benchmarking, we've used an unimpeachable file — not one created specially — a public domain version of the first book of The King James Bible, all fifty chapters of the book of Genesis. This came to **one hundred and forty pages**, well over **forty two thousand words** excluding headers and footers, well over **two hundred and twelve thousand characters** excluding justification ones and **one thousand five hundred and thirty three indexed verses!!** We didn't use a smaller file (as used to benchmark other programs) as PERFECTION SE's timings for most operations then become impossible to stopwatch (too fast!). The hardware used for all timings was GOLD CARD: speeds would be **further improved by over three times** using the ST/QL 030. Of course, LIGHTNING SE was used. File operations were to ramdisk: normal slave blocks would give identical times. All settings on **everything** were for maximum speed, except where indicated to the contrary — we have the sense **not** to force full speed upon you in operations like scrolling and global Search & Replace. PERFECTION SE's speed for these is switchable (at run-time and when configuring), as too great a speed may cause overshoot (with scrolling) or fatal alteration (if there is human error inputting the target or replace strings). Here are the benchmarks for this huge file:

Load 140 pages: 0.6 seconds (yes 0.6, not 6!) ☆ Import 140 pages: 0.6 seconds (yes 0.6, not 6!) ☆ Save 140 pages: 0.5 seconds (yes 0.5, not 5!) ☆ Export 140 pages: 0.5 seconds (yes 0.5, not 5!) ☆ Case-sensitive search from top for word at bottom: 0.4 seconds (yes 0.4, not 4!) ☆ The same, but case case-insensitive: 0.5 seconds (yes 0.5, not 5!) ☆ Case-sensitive search backwards from bottom for word at top: 0.4 seconds (yes 0.4, not 4!) ☆ The same, but case-insensitive: 0.5 seconds (yes 0.5, not 5!) ☆ Automatic Search & Replace, in Fast (No Query) mode, of last 600 occurrences: 7.4 seconds (same length replace string); 7.7 seconds (shorter replace string); 10.5 seconds (longer replace string — longer time as we deliberately chose a high density of replaces to handicap PERFECTION SE into auto-managing memory — without causing any heap fragmentation, but still with only a 0.005 second overhead per replace!) ☆ Automatic Search & Replace in Slow ('Querying') mode: arbitrarily slow, typically 30 times slower — because we deliberately allow for human response time (in case you want to abort) before proceeding from one replace to the next — booby prize to anyone for benchmarking us on this setting!! ☆ Scrolling 100 lines of text, up or down, by full-width screen page: 1.5 seconds ☆ Scrolling 100 lines of text on full-width screen, line by line, in slow (full) mode: 5.7 seconds (down)/5.8 seconds (up) ☆ As above, but in medium speed mode: 4 seconds ☆ The same, but in fast mode and default settings: 13.5 seconds to scroll through the whole massive document, averaging 0.23 seconds per 100 pages (!) — and this could be made up to ten times faster by reconfiguring PERFECTION SE ☆ Reformating paragraphs, changing margins, justification, etc, of existing text: c5 times faster than predecessor ☆ Inserting (or undoing) emphasised, underlined, italics, superscript, subscript, 8 strips, 6 environment settings: Instant (i.e. immeasurable) ☆ Navigation to line or page or to top or bottom or to 8 markers or to highlights/blocks: Instant ☆ Setting new margins, justification, etc: Instant ☆ Deleting block of 100 pages: 0.3 (yes, 0.3 not 3!) seconds ☆ Copying/moving block of 100 pages (not just 10!), downwards or upwards: 3.4 seconds (yes, including all the time for automatic memory management and anti-fragmentation — other programs are light-years behind) ☆ Spellcheck as you type: Ten times faster than anyone can possibly type ☆ Spellcheck all 140 pages in the document using the 350,000 word Mega Dictionary: 3.9 seconds (20 'errors' — like 'pluck!') ☆ And using our tiny dictionary (well, tiny by our standards — large by comparison with most others): 5.1 seconds (566 'errors') ☆ Time taken to create user dictionary from the results of the second spellcheck (566 errors): 0.8 seconds to extract all 'errors' from document and clean document; 1.9 seconds to create a full user dictionary therefrom and also a sorted, duplicate-free wordlist file (for browsing) ☆ Spellcheck file (ASCII or native): Even faster. ☆ Print first 10 pages to file: 3.5 seconds. ☆ Change every occurrence of God to God in bold underlined italics, strip 2 — 9.5 seconds!

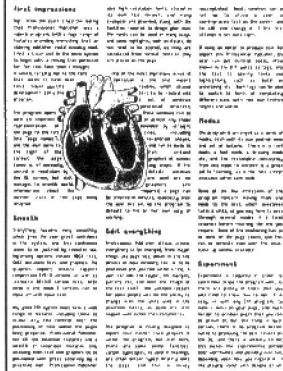
For prices, see the coupon page of our ad. For more info, read our detailed ads in early 1991 for PERFECTION, plus the extra features of the SE (well, about half of them) listed in the June-August 1992 issues. You can upgrade from the standard PERFECTION (or PLUS) to the SPECIAL EDITIONS for the difference in current price, plus £10 (no manuals or dictionary disks to be returned — we'll send manual supplement). No discount to users of other word processors.

QUICKLASER — The definitive output tool from PRO PUBLISHER to HP LaserJet II (or compatible) printers. Printed output quality subjectively exceeds that from any other QL product.

TRANSFER UTILITY SPECIAL EDITION — Does everything — 16 case change options, 14 types of sorting (multiple sorts possible), auto string translations, etc.

LIGHTNING SPECIAL EDITION GOLD CARD VERSION — See June-Aug 1992 QLW for details: optimal speed from GOLD CARD, ST/QL, THOR XVI. Free upgrade from the ROM SE version (return ROM + disk) if you are ordering something else at same time: if not, £10 charge.

Professional Publisher is the best! No other desktop publisher for the QL even comes close.



C68 C Compiler Source

When V3.00 of the **C68 C Compiler** was released, in June 1992, only the runtime disks were available. The Compiler has now been upgraded again, and the source code is now available.

The Compiler is now version 3.01; some bugs detected in 3.00 were quickly put right by Dave Walker. The source code has been compressed onto two 3.5 inch 720K disks. Disk 1 contains a boot program which will load the Packer/Un-Packer, a program called RamCopier for users with single disk drives and two of the compressed source code files. Disk 2 contains the remaining compressed source code files.

C68 V3.01 Runtimes (3 disks) £3 (pre-copied disks), £1.50 (user supplies disks) postage & packing 50p (UK). C68 V3.01 Source Code (2 disks) £2 (pre-copied disks), £1 (user supplies disks) postage & packing 50p (UK). C68 V3.01 Full Issue (5 Disks) £5 (pre-copied disks), £2.50 (user supplies disks) postage & packing 50p (UK).

Orders and enquiries to **QubbeSoft P/D, 38 Brunwin Road, Rayne, Braintree, Essex CM7 5BU. Tel. 0376 347852.**

Dutch Translation Help Wanted!

Qubbesoft's Ron Dunnett has a challenge - he wants a QL user to help translate Dutch software for the public good. "I have kindly been given loads of very good Public Domain software for the QL by the Dutch QL User's group," he writes. "The trouble is that most of it has Dutch text or .doc files, so I can't put it out to English users, as they won't have a clue what's going on (unless they can read Dutch). Some of the programs are very good, and deserve a much wider audience."

Plus, says Ron, success might stimulate the authors to write more software in the future. He also has German software that has its text files which need to be translated into English from German. A previous offer of help from a user in Switzerland fell through when Ron sent him the files and heard nothing further!

"I'm getting more and more software from Europe, for my public domain library," says Ron, "And very little from UK users. I find this very odd."

Contact QubbeSoft P/D at 38 Brunwin Road, Rayne, Braintree, Essex CM7 5BU. Tel. 0376 347852.

Zikes! A Small Delay

The Zike, Sir Clive Sinclair's latest invention, is reported to be running behind its original production schedule.

The electrically-assisted bike, which Sir Clive hopes will recoup some of the losses sustained by his previous vehicular venture - the C5 electric car, unkindly compared by some to an electric bath-chair - was promised for the end of May. Customers who sent cheques for the £499 bike have been sent letters of apology, and will be given a discount when their machines are despatched.

The Zike is now expected to be in full production in August. With a top speed of 15 mph, licence-free road use by anyone over 14, and a non-polluting power source keeping it 'green', the Zike may be heading for success. The failure of the C5 was blamed by many on unreliable batteries - slow speeds, users say, are no problem - but no speed is. Battery technology has advanced in the intervening years.

Sir Clive blamed the delay to the Zike on problems with a production tool. The signs are, though, that Sir Clive's fans don't mind waiting a little longer. "I'm sure the Zike will get here," said one. "I just hope he isn't going to try and invent the Wheel first."

Dried to Distraction

By the way, thank you to everyone who wrote to console us for the troubles we lived through earlier this year and saying how pleased you were to see us back. *QL World* felt a bit like a ghostly swan during that time - nothing showing above the water, frantic paddling underneath!

Apologies also to those who were baffled (or entertained) by some strange expressions in Mike Lloyd's review of the Gold Card in the May/June issue. For that issue, we read nearly all the copy through an optical scanner. It was then swept with a spelling checker (note well - a very bad combination, if our experience

is anything to go by) and then properly proofread.

Unfortunately, under the pressure on our first Production Day, the pre-proof version of the file was imported onto the page.

All *QL World* features are now read by the editor before they go onto the production computer. (With one or two small but notable exceptions which are still giving us gyp in the holiday season.)

Meanwhile, we are still trying to work out what a hue memory is, how you roup a file, and why anyone should want to dry a disk. Suggestions to Open Channel, please.

WINNING THE POOLS!

Businessman Stan Caton has pooled his QL resources in the hope of a win. David Drysdale reports.

When North Manchester businessman Stan Canton, now semi-retired, bought a QL in the machine's early days, it was not a toy he was seeking but a professional machine capable of some very hard work.

The work began the day the QL arrived, but it was Stan himself who had to do it as he laboured to program the machine with the help of the manual. His wife Maggie quickly became a computer widow as he would sit at the QL, on his days off, from 5am until late evening and would have to be physically shaken out of his concentration to come away for a meal.

There was one bonus for Maggie, though. Stan had always been an early riser who would waken the family by walking around at 5am. When the QL came he would go quietly to the keyboard as soon as he awoke and Maggie would get some peace.

QL housing

From the start the QL and its peripherals took up too much workspace, so Stan designed a timber unit to keep it all together on the table top. It was probably one of the very first 'workstations' ever built, but Stan named it the Compu-cabinet and offered it for sale in *QL User* at £45. Only four people sent for it and so, naturally, Stan stopped making it. He still uses one of the cabinets which he has mounted on a foldable metal framed trolley so that it can be transported to demonstrations and exhibitions.

Stan's next venture on the QL was the design of a business cash flow chart which he developed with the aid of Maggie who was a professional bookkeeper. Again he advertised it with *QL User*, and this time he was more successful. The program had been intended in the first place as an aid to Stan's own businesses and much impressed his bank manager. The *QL User* sales were a welcome spinoff.

Stan's varied business ventures included professional photography, kitchen designing and running a launderette, but whatever he was doing he found the QL was at hand to help. When selling kitchens, for example, he designed a program that

would not only assist in designing the kitchen layout but cost up the job as it went along.

This program so impressed a business colleague who was into bedroom furniture manufacture that he sought Stan's help in developing a way of computerising his own price lists.

Adaptable Abacus

In his programming work Stan finds that Digital Precision's *Editor*, along with the *Turbo* compiler, is an absolute must. Of the supplied software he is most impressed by *Abacus*, which he finds is amazingly adaptable and can organise business and accounts applications of every kind. He is also quite fond of *Quill* which he feels has been given a new lease of life with the Gold Card. He now uses front end program *Qpac2* which, when eventually mastered, revolutionised his program and file handling. A PC-style keyboard completes his equipment.

An important development for Stan has been the setting up of the Manchester Quanta group QUM. Prior to this everything he accomplished on the QL had been done the hard way by having to work through his problems alone and with only the manuals to help him.

The group is now involved in importing public domain software from PCs into the QL and Stan has been quite busy assisting Sohail Bhatti of QLaw and the other members with this work. Again, *Editor* and *Qpac2* have come to the rescue and Stan has written a program to automate processing of a full disk of files.

Washed out

All this QL activity is having a much needed therapeutic effect on Stan who some time ago had a traumatic time with the collapse of one of his businesses. The business, a very popular launderette on a council housing estate had been so busy that Stan decided to re-equip it with the latest and best machinery.

This was no sooner done when the council handed the estate over to private developers who modernised the homes

and installed washing machines in all of them! Not even the best of Stan's sophisticated QL cash flow programs could save him from the financial meltdown that followed.

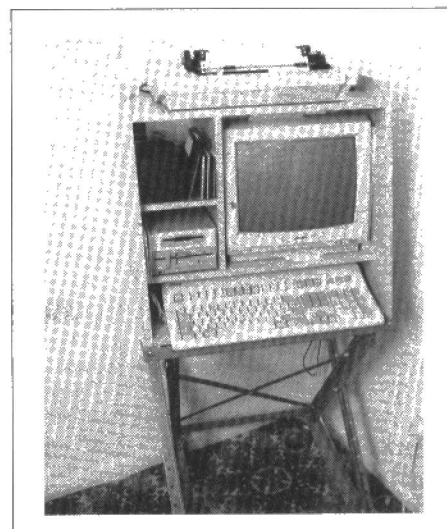
Stan, however, seemed quite unperturbed by it all as he led me from his luxurious lounge overlooking a small forest of trees to his QL den to see his latest brainchild - a Football Pools odds-analysing program. There was nothing silly or mystical about it, he assured me, as it was based on mathematical analysis of the methods of the only real pools experts, the Bookies, with the sole aim of increasing the winning odds.

Fixed odds

It is quite a sophisticated program and its development so far has convinced Stan that while it has a huge potential for increasing a punter's likelihood of winning the Treble Chance it is a must for enthusiasts who favour Fixed Odds coupons. They, he assured me, are likely to have even better results and should see a steady profit from their pools investments. He has no intention of setting up as a pools tipster, he said, nor was he inviting the public to join him in some gigantic permutation. His aim was to develop, refine and debug the program which would be offered for sale in a *QL World* advertisement. The punters would then be given a powerful tool with which to develop their own forecasting skills.

As Stan, Maggie and I sat down with a cup of coffee Stan reminded me that the program had really been intended for his own use and he had great hopes of landing a good win with it. It would also be on exclusive offer, at the start, to owners of the QL.

Just as there are people who have decided to stick with the QL because of programs such as *Perfection*, there will be others, I suppose, who will find that Stan's new program will give them all the incentive they need to stay with the QL - the country's "winning" computer.



QTOP

Del Allen tries out the most recent version of QTop for the QL and Thor.

INFORMATION

Program: QTop

Publisher: Cowo, Munsterstrasse 4
CH-6210 Sursee, Switzerland.

Available from: Software87

33 Savernay Road
London NW3 2JU

Price: £29



QTop is a front end for the QL. A front end usually consists of a series of menus from which you can select various options intended to make life easier. For example, selecting Files will result in the appearance of a sub-menu offering various things to do with files, such as View, Delete or Execute.

When QTop is booted, it helpfully decides which of its files to load, after checking the rom type and the amount of memory. The user merely has to insert the disk and press a key to inform the program which computer is in use - QL, Thor, Amiga etc. After a short wait the Main menu appears. Here is an option to alter the default devices. For example, one can specify that, say, flp2 is to be used for data and flp1 for programs. Another option calls up a jobs manager.

Here one can do such things as listing all jobs, suspending a job, and so on.

Yet another option is Myprogs. This will present you with a list of programs, provided that you have previously created the list. The method of creating a list is somewhat inconvenient, and the list itself is of doubtful use. The Tools option presents a few basic utilities such as hex to decimal conversion. Options allows changing baud rate among other things.

Unforgettable

Desk puts you into the Desk menu. Here you can remind yourself of which solid colours are available. If you have forgotten the colours in mode4, for example, you will see four bands of colour viz black, red, green and white. This is not something I forget very often. System-Info will tell you things like which rom you

have. Mem-Info lists a few of the Qdos system variables. Clocks presents a choice of clock types, both analogue and digital. These are fun to look at the first time. A large analogue clock face can be shrunk and placed in a corner of the screen. Some people may prefer this to the digital displays - I did. But I normally use a clock provided by Toolkit II.

Snap is a screen dump utility. I have not tried this. QTop makes heavy use of the QJump Pointer Environment, which is a part of the package. The fact that the pointer environment is on the disk makes the package reasonable value for money. One could be forgiven for thinking that if one already has the pointer environment, there would be little point in purchasing QTop, but it has a few features that

Pointer Environment then consider buying QTop.

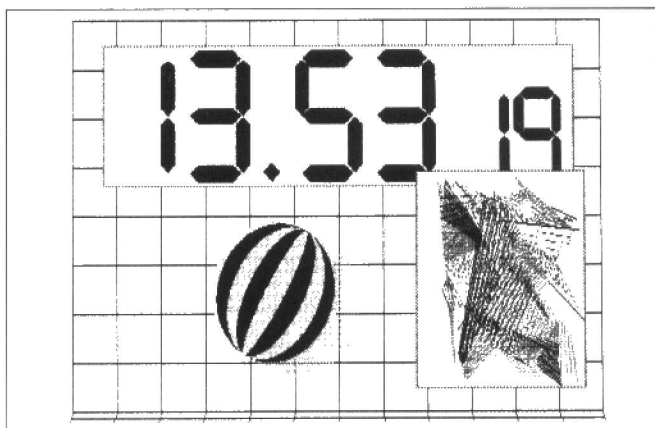
The manual is very well written considering its foreign origin - it is much better than many English manuals, though it lacks an index. As a bonus it includes details of the hotkey and TKII commands. It is extremely pleasant to have these all together under one cover.

The rest of the QTop package should be regarded as a bonus. There are a lot of demos, intended, presumably, to impress with their speed. Sadly, visually, they are largely unimpressive. The sight of coloured blocks, for example, appearing and disappearing with extreme rapidity very quickly becomes boring and will not impress, say, Amiga owners. One or two, though, are quite pretty. The animation demos are a bit uninspiring, as one cannot do anything with them other than watch. But they are not part of the Qtop system and should not be criticised too heavily!

Different speeds

The last item, Tindex, is different. It purports to show differences in speed in various operations between a standard QL and your system, and as such is interesting and informative. It shows, for instance, that a QL with a 512K Expandram is slightly faster than a 128K QL, as one would expect. I was surprised to find, however, that disk handling was significantly slower. Upon querying this with the author, I was informed that Tindex should be run on an 'empty' machine. It did not and does not say this in the manual. To run Tindex on an 'empty' machine it is necessary to load up an extension file. I have copied Tindex and the extensions to a separate disk and have had a lot of fun trying different *Lightning* settings. But Tindex still tells me that disk handling with the Expandram is slower than a standard QL with microdrives!

To sum up: forget about the demos - they are free, anyway. The QTop system itself is worth having if you need a front end and do not already have the Extended Pointer Environment. A small proportion of its features are of limited usefulness, but the system as a whole may make a useful addition to your roster of working programs.



4. QTop-DESK

QTop-Desk v1.04 © 1989 by COWO			
JOB	Job tag	owner priority	8 Jobs running !
1ST JOBS	<	<	CANCEL > page 1 >
JOB-INFO	0	0	532 SuperBASIC
WIN PRIOR	0	0	5127 HOTKEY
PRIORITY	0	0	527 TDESK
REMOVE	0	0	46 TELCKS
SUSPEND	0	0	50 TDESK
RELEASE	0	0	50 TINDEX
OLD	0	0	50 TSNAP
WAKE	0	0	50 TSNAP
NEXT JOB	<	<	ALL CANCEL > >
MONDAY 1991 Jan 07 12:19:25 278528			

This is QTop-DESK your personal system manager !

make use of the pointer environment easier and this alone, I believe, makes QTop a reasonable buy.

For instance, QTop somewhat eases button handling. A button is a little window with a program name in it. The button can be easily created and can be used to put a program to 'sleep', wake it up, and so on.

Curate's egg

As I intimated earlier, while there are some features that I could live without, QTop also has its good points - it's a real curate's egg in fact. If you do not already own the Extended

OPEN CHANNEL

Open Channel is where you have the opportunity to voice your opinions in *Sinclair QL World*. Whether you want to ask for help with a technical problem, provide somebody with the answer, or

just sound off about something which bothers you, write to:

Open Channel, Sinclair QL World, The Blue Barn, Tew Lane, Wootton, Woodstock Oxon. OX7 1HA

Pirate Min!

Could you please let the sceptical know that the pirated Minerva does exist - I bought it at Münster! They guy selling it was apparently too uninformed to realise it wasn't genuine. It was a very early 1.66, and didn't even work without much tweaking. It was no April Fool.

On a slightly different tack, the eeprom in the MKII Minervas (the real time clock, etc.) is identical to the MKI. It is worth pointing out that the Minerva clock is only connected to the QL by a serial bus. The protocol for changing

this clock is extremely complicated, and QL crashes will not corrupt the clock, unlike all other QL clocks, including the memory-mapped Gold Card clock. In fact, we recommend users of both to unplug the Gold Card battery. It is possible to crash the Minerva clock when the device is being transported outside the QL. If yours becomes corrupted, then EXEC_W the program MINICONFIG_OBJ to reconfigure it. Minerva MKII needs version 2.26 of the Gold Card, or greater, to work properly with it.

Tony Firshman
TF Services
London W2

Driven

I thought I had printer drivers licked. My printers have been working fine for several years now. Then, for quickness, I decided to print a draft copy of a two-page document from *Quill* onto my Brother M1109 printer. Horrors: each time the download to the printer was completed, it printed the last few lines of the page, and then continued with the rest of the document. Also the out-of-place lines were offset to the left, ignoring the margin setting.

I've tried juggling the printer codes in the driver, such as making the page length shorter than that set in *Quill*, but to no avail. Can anyone suggest what the problem is? Since I mainly use my DX-100 for printing documents, it will not hinder my work, but I am curious as to why the M1109 refuses to print multi-page documents without messing up the print.

James C McGreehin
Alva
Scotland

Editor's comment: This is just the sort of query that, were all our files in order and the many kind offers of help with printers that we had last year easily to hand, I should be able to pack off to a responsible party. For now, any responsible party with a answer for James, please write to Open Channel.

Unblocked

Referring to my article *Three in the Hand*, over which there was some controversy, I should explain how I arrived at my block-move timings for the three programs. I was using version 2 of *Perfection* at the time I wrote the survey. V3 is faster, but this is not the main point.

John Bunce (*QL World* July 1992) was wrong - I did not forget

any decimal points. *Perfection* can move massive blocks of text for miles and miles almost quicker than the average giraffe can blink. Nevertheless, it took this user, to the nearest second averaged over three similar attempts, 25 seconds **of total user time**, not just the text-move itself, to move one paragraph one page lower. All three programs, as the article stressed, performed more than adequately.

This is the process I used: I began timing with the cursor at the start of the block. With *Perfection*, you highlight the following text, move the cursor to the end of the block, mark the end of the block (by unhighlighting), define the highlight area as a block, move the cursor to where you want the block to go, and then move the block. This takes time. To avoid clumsiness, I practised with each word-processor, and the final figure is the average of three efficient tests.

Michael Edwards
St Mellons

*Editor's comment: Since this is the type of program to which *Perfection* is often compared, I tried the same move on one of our 286 machines running the most recent version of a well-known MS-DOS wordprocessor.*

Going through the same sequence - that is, starting with setting the block - to move a 10-line block from top to bottom of a large file took 55 seconds.

This was longer than I had expected! And then I had to move it back again. The bulk of the time in this example was the actual file-moving.

To move a four-line block down ten lines took about 15 seconds. In this example, the file-move time was virtually instantaneous.

You can see by comparing these figures what proportion of the time (about .25 of a minute, assuming the user can see exactly where the text is to be placed and does not have to go looking for it) was taken up in setting the block and moving the cursor to the destination position.

Editor's notebook

Activist Ron Dunnett (Qubbesoft, Quanta) has been looking for help in translating the text of German and Dutch public domain software into English, (see *QL Scene*), but so far even such offers as he has had have not borne fruit. It seems a pity that these programs should be under-utilised because of language barriers.

We hear that Sir Clive Sinclair's Zike bike is a little behind its production schedule. If it runs true to form, it should still have a strong following in a number of years, just like the Spectrum and the QL.

A gentleman in Scotland who reads *The Beano* compares our correctness unfavourably with that venerable journal. I should hope so. The Beano's publishers are fierce in defence of their honour, and I would be proud to know the proper form of the word OYAAH, but my dictionary, alas, doesn't seem to include it.

Spare a kind thought for Mike and Sue Lloyd, moving house (they very much hope - "we are still on a knife-edge", says Mike) in these troubled times. And a thought for all the folk who can't, too.

Perfection uses one more process (defining the block) than my test-piece, but moves the block faster over long distances. Another well-known MS-DOS wordprocessor, for instance, moves its blocks faster than my test-piece, but also has a longer defining sequence. You wins some, you loses some. This is the ballpark that a powerful word processor is operating in.

This is a realistic way of assessing block-handling. Although fast file-handling is one of the greatest blessings, in a complex operation like block-handling, the text-marking and locating times should be included, as has been done here.

There have been updates of two of the three programs surveyed in Mike's user report since he wrote it. All three programs were versions which were on sale to the public and being used in earnest, and not beta-test or preview models.

Busy Wait

I have seen the article *Systematic Machine Code Programming* starting on page 45 of the April *QL World*. I was surprised to find Alan Bridewell advocating a busy wait, that is, a tight loop doing nothing with, I suppose, the intention of allowing other jobs to use the processor! The job doing the waiting is, of course, still using cpu time, which other tasks might want. Also, the delay depends not only on the number of times the loop is executed, but on the processor speed (Gold Card, etc.), memory expansions, what other tasks are running, and so on as well. What he should have been using is the Qdos call specifically for this purpose: MT.SUSJB, which suspends (pauses) a job for some fixed length of time allowing other jobs to run. Something like this:

```
.keyrow
moveq #58,d0 ; Code no. for MT.SUSJB
moveq #1,d1 ; Suspend this job
moveq #2,d3 ; Time in units of 1/50 sec
moveq #1 ; Do it
; Now the keyrow routine proper
lea keyr,a3 ; Keyrow param table
```

And so on. A similar change should be made to the other two

routines. Also, in the keyrow routine as printed, the loop jumps back to initialising the counter rather than the NOP instruction, giving an infinite loop.

Ian Jackson
Ponteland
Northumberland

Alan Bridewell replies: Mr Jackson is quite correct. The best solution is to suspend the job to prevent excessive calls to the IPC. Why I didn't think of that at the time is beyond me - my only excuse is that I am a self-taught programmer, such mistakes are bound to crop up eventually. However, I must add that the tight loop does work. Although it wastes valuable processor time, it wastes far less than accessing the IPC several times in one keypress! And if it works, I tend to use it. But in future, I shall use MT.SUSJB.

He is also correct in pointing out that the Keyrow routine as printed does not work. Unfortunately, I sent an early (bugged!) version to QL World for publishing, instead of the final version. Please accept my apology for this. Note, however, that the sample program which uses the routines does work, because it contains the correct version. I do try to test everything. I actually compile the output from the word processor for the articles, to check the programs! This, of course, won't work for the separate routines, because even if they compile, they won't run as programs.

The Editor adds: Alan has provided two sheets of updated listings. Users who would like a copy of these listings, please send an SAE to the Arcwind office and I will arrange for copies to be sent.

Health

Mr. Freddy Vachha of Digital Precision has been in touch with me and spent a long time trying to analyse a fault which has been giving me trouble with some programs. We have jointly come to the conclusion that it must be either my disk drives, or the Trump Card/ interface which is at the root of the problem. He has asked me up to his office together with my 'kit' for a health-check/MOT (for the kit, that is), hoping that we may once and for all sort out the problem. I shall take up his offer when time allows.

Michael J Grover
Leatherhead
Surrey

Early Model

I own an early Sinclair QL, serial number D12-063456, and I understand that this machine may have problems with some printers. Can anybody recommend to me a good printer which will work with my machine without too much messing around?

K W Blackmore
Stamshaw
Hants

Chess

From time to time I have read that Psion Chess will not work with a memory expansion. The latest mention appears in *QL World* in Mike Lloyd's review of the Gold Card in the June 1992 issue.

I have a 640K QL, and I find that Psion Chess operates perfectly using the boot program listed here. This boot program must be on the same medium as a copy of the file 'chesse' supplied by Psion. The original mdv cartridge must, as always, be in mdv2 at run time.

Mdv users should change flp1 to mdv1 in line 160 of the listing here.

No special toolkit is required, and the system runs satisfactorily on JM and JS roms; probably on others, too.

C B Storey
Tyne and Wear

Comment: Thanks to Brian and other Psion Chess players who commented on this.

Boot program for Psion Chess

```
100 MODE 4
110 OPEN
#1,con_512x256a0x0_32:CLS
120 CSIZE #1,1.0
130 CSIZE 3,1:AT
2,12:PRINT"QL CHESS"
140 addr=184064
150 a=393216
160 a=RESPR(a-addr)
170 LBYTES flp1_chesse,addr
180 CLOSE #0:CLOSE
#1:CLOSE #2
190 CALL addr
200 STOP
```

Head Left!

Referring to A R Kempton's problem with his printer: to get the printer head to return to the left hand end of the printer, he must

include the following in his printer_dat:

PREAMBLE ESC@
POSTAMBLE CR

Those are, respectively, a printer reset and a carriage return.

The latter is required because, in Quill, if he leaves the cursor at the right hand end of the last line, then the last line will not be printed but will appear at the top of the next document he prints.

If, however, the Taxan printer is not Epson-compatible, he must read the Taxan handbook to get the appropriate equivalent codes.

A Owen
Stowe by Chartley
Stafford

Re A R Kempton, Open Channel, *QL World* July 1992, he should check that his printer driver has a CR character (code 10/10) included in either/both of the PREAMBLE and POSTAMBLE strings.

Chas Dillon
Tamworth

Like your subscriber A R Kempton, I had trouble with my printer (a Brother HR-15) not returning to the left-hand margin at the end of a document, and then starting the next one in the wrong place.

After exhausting my meagre store of technical knowledge to no effect, I decided to try guile.

My first cure was to put a full stop on a new line at the end of each document, and this of course caused the carriage to return there, but now I just leave a blank line at the top of each document and, wherever the carriage is, it moves back to start the second line as shown on the screens.

For the same problem with printing from Archive Database, the alternatives are to incorporate LPRINT as the first line of each PROCEDURE, or when necessary to type LPRINT "" before calling the procedure you want.

I hope this helps.

T Rendall Davies
London N21

Comment: Conventional and less conventional solutions there, which I hope will give assistance in both directions. The trouble with guile is that it can provoke counter-guile from your machine in some unexpected places.

Abacus: only following orders

In Part 2 of his Abacus spreadsheet, P H Warne shows how to deal with payments that don't occur every month.

Last month you started laying out the spreadsheet which promises to solve all the problems you have with your bank account - except how to keep it topped up!

The formulae arriving at the latest payment date for quarterly and biannual payments proved the most difficult to compile. The principle used is to calculate the offset from a 3,6,9,12 sequence for quarterly payments (or a 6,12 sequence for biannual payments) for both the current date and the standing order date. The first offset is then deducted from the current month and the second added to it. A further 3 (or 6) is deducted if the second offset is greater. The complications arise when that takes you into the previous year, and Abacus comes up with a date like 1991/00/03! It took a long time to sort that one out.

Table three gives the formulae to be entered. Two more constants for the current date are added to the table in column AI - the offset of the month from a quarterly and a six-monthly sequence. The formula in column C calculates the offset of the first payment date, adding 3 (or 6) if the offset is zero and the number of the current month is less than 3 (or 6). Column E assigns a value of 9 (or -6) if conditions are such that the most recent payment would be in the previous year, 3 (or 6) if the offset in column C is more than that for the current date, or zero. As with the monthly payments, both columns assign a value of 15 if the current date is outside the limits one side or another. Column F combines the offsets with the number of the current month, subtracting 1 from the current year if necessary, to arrive at the date of the most recent payment. The formula in the monthly columns I to AG is the same as for monthly payments, but it is only entered in those columns which would apply to that particular payment. For example, for a quarterly payment first due in February, only the monthly formula in the February,

TABLE 3 - variants for un-expanded QL

"Flag" column D omitted.
Date Constants will be in column AF.
Formula in H3 (month headings) etc., will be:-
month((col()-6)/2)
Formulae in columns C (Quarterly only) & E (Latest Payment/Debit) will be:-

Monthly:-
if(\$AF4<days(Bxx),Bxx,if(\$AF4>=days(Dxx),Dxx,Bxx(to 5)+
str(\$AF5-if(\$AF6<val(Bxx(9 to)),1,0)+100,2,0)(2 to)
+Bxx(8 to)))
Quarterly:-
In column C:-
if(\$AF4<days(Byy),15,int((val(Byy(6 to))/3-int(val(Byy
(6 to))/3))*3+0.1))
In column E:-
if(Cyy=15,Byy,if(\$AF4>=days(Dyy),Dyy,Byy(to 5)+str(\$AF5
-AF7+Cyy-if(Cyy>\$AF7 or (\$AF7=Cyy and \$AF6<val(Byy(9 to
(6 to))/3)+100,2,0)(2 to))+Byy(8 to)))
Bi-Annual:-
if(\$AF4<days(Dzz),Bzz,Dzz)
Annual:-
Bnn
Formulae in Monthly columns will be:-
if(\$AF4>=days(Bxx) and val(Bxx(6 to))<=Mn and val(Exx
(6 to))>=Mn,Fxx,"-----")
where Mn=number of month

If weekly credits/debits are required, the formula in column E is "Bxx" and the formula in column H is amended.

May, August and November columns should be copied.

Weekly intervals

It is also possible to cater for credits or debits due at weekly intervals, or at intervals in multiples of a week, but these are treated differently. For simplicity, the total amount covered by the period between the date of first payment and the current date is shown in the January column. Although it is possible to break this total down into monthly sums, the formulae are all very long and in some columns Abacus has to be forced to accept them by subterfuge! It was felt that the end did not justify the means. The date of the first payment in the current year is calculated in column F. The two formulae are shown in **Table four**.

When the basic spreadsheet has been drawn up, any of the formulae for regular payments which are also required

in the Credits section can then be copied up. In **Figure one**, you will see that this has been done for the monthly payments. The whole block A17:AG0 was copied to cell A5 and the text in cells A5, F6 and A7 edited. However, biannual credits must not be above row 9, to avoid forward referencing.

The next thing to do is make a list of all your standing orders or direct debits, and the dates and amounts involved. Some standing orders will need two or more rows allocated to them. For instance, as **Figure two** shows, Poll Tax starts with two payments from the previous year's assessment, then a single payment in April followed by nine others for a lesser amount. The second row shows the same date in both columns for the payment of the initial sum and the third row shows the dates of the first and last subsequent sums. Almost certainly, more than one row will be needed for mortgage payments to allow for changes in interest rates.

Accommodation

To accommodate all these payments, expand the spreadsheet by inserting as many rows as are required for each category of payment (or credit). Note that rows are inserted above the cursor position or the row number typed. To do this, place the cursor in the desired row and type F3 G I Enter nn Enter Enter. All the formulae are adjusted automatically, so provided the summation formulae include the sum underlined row, any inserted rows will be included in the totals calculated.

When the rows have been inserted, use the Copy command to copy the data and formulae in the top row of the section to the row below, by typing F3 C Bn:AGn Enter Bn+2 Enter), repeating the process as often as necessary to fill in the inserted rows. Then fill in column A and enter the correct dates and amounts in place of the dates so copied. For payments other than monthly, the formulae in cells Ixx to Agxx are only copied in the columns where a payment would fall due, and not across the whole row. **Figure 2** shows how the finished product should look.

Because the Insert and Copy commands make it simple to insert a row when needed, there is no need to leave empty

rows to allow for future changes.

Cash withdrawals and payments by cheque or card transfer should be entered in the appropriate monthly columns below the standing payments. It is easy to get into the habit of doing this at regular intervals. It is not necessary to calculate the balance on every occasion.

Five seconds a row

Calculation, using the F3 X command, takes a few minutes (about five seconds per row for those with the full standing payments formulae). On typing F3 X, you are asked for the date for which the balance is required. This is in the usual English format D(D)/M(M)/YY, which is converted for Abacus use in cell AI3. To view the balance go to cell AExx in the last row (F5 AExx).

The formulae in the totals cells make it possible to obtain 'end-of-month' balances for any month, ignoring credits and debits which have been entered for subsequent months. This makes for easier comparison with bank statements, which are usually sent out at the end of the month. In fact, it would not be too difficult to add Nigel Bates' reconciliation procedures at the end of the sheet, to make comparison with the bank's statement even easier. Make a list

of the cell references for payments not shown in the bank's statement, and type these references into the appropriate section of the reconciliation.

At the end of the year, it is only necessary to change the year in cell B2 (remembering that this is text), transfer the balance calculated for 31 December to the 'Balance Brought Forward' cell (I4) and delete any rows covering payments which ended in the past year and are no longer relevant.

Limited ram

For those with limited ram, all dates must be quoted in terms of the current year and only the twelve monthly columns used. Flag column C is retained but only used for quarterly payments. The formula in column E showing the most recent payment is altered accordingly. In the monthly columns, beginning at H, the comparison need only be between the number of the month and not the days of the whole date. The main alterations are shown in **Table five**. There will also have to be adjustments to text in some places to fit into the reduced space.

Memory can also be saved by deleting any parts of a row beyond the last month in which a payment will appear, that is, beyond the month of the final

1	BANK ACCOUNT #12345678 - A.N.Y. BODY	For current Balance go to cell AE45																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																												
---	--------------------------------------	-------------------------------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

payment. Since that row will be deleted entirely at the start of the next accounting year, no problems will occur when the new year is opened. Doing this will also speed up calculation a little.

If the whole spreadsheet is printed out in condensed type (137 characters per line), it will occupy three A4 sheets. With the monthly text columns only 10 characters wide, there is a margin on the right

hand side of the third sheet. There is, therefore, scope for increasing the width of these columns without taking up more paper, but the information on-screen at any one time would be reduced.

ACCOUNT #12345678										DATE CONSTANTS:									
21 ACCOUNT FOR YEAR 1991										1991/09/26									
31 DATE:- 26/9/91										149287									
41										9									
51 STANDING CREDITS										26									
6 MONTHLY:-										3									
7 OTHER CREDITS:-																			
8																			
9																			
10																			
11																			
12																			
13																			
14																			
15																			
16																			
17 STANDING ORDERS & DIRECT DEBITS																			
18																			
19																			
20																			
21																			
22																			
23																			
24																			
25																			
26																			
27																			
28																			
29																			
30																			
31																			
32																			
33																			
34																			
35																			
36																			
37																			

Figure 1 - the basic spreadsheet.

Rom Switch from QLEA only

"A small but very significant error" has crept, according to Quanta, into July's "otherwise excellent" report of the Bristol Quanta Workshop.

The Minerva/QL Rom Switch was mentioned in the report as being available from Qlas.

Syd Humphries stresses that the Switch was designed by, and can only be obtained from, QLEA, the Quanta East Anglia Subgroup.

The Switch costs £25 plus £1 postage (£2 outside the UK and Europe), from **QLEA, c/o Geraint Jones, 1, Bridgate Court, Thetford, Norfolk IP24 3AY, UK.** Payment by sterling cheque to QLEA.

Syd sends his best wishes to *QL World*.

Quanta organiser solves two in one

A Quanta workshop is being planned in Kent for 5/6 September 1992. The Venue is the Swan Hotel, Ashford Road (the A20), Charing, Kent. Tel. 0233 712357. Overnight accommodation is available there and nearby. Bar and refreshments will be available. The organiser is **Paul Merdinian, Forge House, Charing Heath, Ashford, Kent TN27 0AX. Tel. 0233 712857.**

Paul is launching his program *QL Super Crossword Solver* on the Mardi Computers label at the workshop. Supplied on 3.5in disk and designed to "work on any QL, even a 128K base model", the program "works like electronic spellcheckers, accepts input of known characters and ? for unknown, and utilises controlled scrolling." The software also apparently includes "Forty programs, twenty in SuperBasic, for you to hack to your heart's content, and 20 superfast stand-alone compiled versions. Contains over 20,000 words." The program costs £10 inclusive of UK postage. Enquiries to Paul Merdinian at the address above.

SOFTWARE FILE

QDESIGN

INFORMATION

Program: Qdesign 1 running under the Pointer Environment (included).

Publisher: Jochen Merz Software, Im stillen Winkel 12, D-4100, Duisberg 11, Germany. Tel. 010 49 203 501274.

Price: Qdesign 2 recently released: £48, including laser printer driver. Upgrade from Qdesign 1 £13. Runs with rom version higher than JM, and Minerva.

A F Wilson became a Qdesign user last November, and sees it as a route to advanced graphics on the QL for more recent roms.

The graphics/art market is surprisingly small for a mature product like the Sinclair QL. In the early days, software like *Eye-Q* (Digital Precision) and *Artlee* (for those running the Ice pointer environment) dominated the market. However, neither of these programs can now offer the power, sophistication and freedom of expression of *CorelDraw/Dpaint/Harvard Graphics* for the PC, or *Dpaint* for the Atari ST/Amiga. However, the continued development of quality European graphics software - *Painter* (Progs of Belgium) and now *Qdesign* (Jochen Merz Software of Germany) will challenge established QL graphical software and bring the QL closer to the mainstream. Qdesign is advertised as a 'Graphic and Design Program', which suggests that Qdesign has aspirations in the DTP market. Unlike its graphics presence, the QL scene is abundant with powerful DTP programs like *DTP/DTP SE* and *ProPublisher* (Digital Precision), *PD2/PD2+* (Dilwyn Jones Computing) and now *Qdesign* from J Merz Software.

Within 10 days of ordering, Qdesign arrived from J Merz Software (Germany). The package included a short but very informative A5 manual (34 pages) and a program disk (3.5 in). The software is not copy-protected, which has to be applauded, and backup was straightforward (TK2 wcopy command worked fine). The boot program was run from disk, first, as if back in time to the good old days of the ZX Spectrum. The Qdesign loading screen quickly appeared with the obligatory Space Shuttle. The program finished loading and the eye-catching Qdesign main page appeared.

The first thing you notice about Qdesign is the Qpac2 pointer environment (by Tony Tebby of Qjump). I must admit to being a big fan of the Qpac2/Qimi mouse/pointer environment. The screen layout of Qdesign is spacious with smart icons and easy-to-use menus.

At this point, I will diverge a little. The Qimi mouse board with clock (from Quanta or from J Merz Software), connects onto the main QL motherboard. The instructions are clear and installation straightforward. However, the first batch of Qimi boards gave some QL owners

problems - the board pin connectors are a bit on the short side, and with some QLs, the board has a tendency to spring out. Quanta, who make the board, were alerted to this fact and have fixed this for the second batch. Qimi is Gold Card compatible. Qimi works with any Atari compatible mouse (I picked up Truemouse for £14.99 at the Glasgow All Formats Fair last year). I highly recommend the Qimi mouse and in conjunction with Qpac2 (now discounted from Care Electronics) results in a powerful and friendly WIMP-type environment.

Pointer extensions

Qdesign includes some Qpac2 pointer environment extensions, and therefore does not require Qpac2 to work. Qdesign works equally well with both QL function/cursor keys and/or the mouse. The mouse makes art less restricting and allows for greater artistic expression. The cursor is an arrow, and it is easy to locate on screen. To get started is simple: just start drawing on the sketchpad window, selecting icons from the Tools bar as needed - you have a choice of spray gun, text, paint can, shapes (square, circle and triangle - triangle includes other polygons, such as rectangle, pentagon, hexagon, etc.), different line patterns, magnification, and so on. All operations were instant. Clicking on the scrollbars at the bottom (left/right) or right of the screen (up/down), scrolls the sketchpad window over another part of the page, however, the scroll bars could be a little wider in the y-direction. Selecting F5, or the Page icon, previews the entire page in low resolution (similar to Propublisher). F10 gives high resolution page preview.

The top line of the menu bar includes a number of Qpac2 icons, like putting Qdesign to Sleep, as a button (a button is a non-graphic icon) or Resize Qdesign or Move Qdesign. The bottom line of the menu bar, is the options menu Disk, Print, Font, Options, Block, Status, Blobs, Help and Quit. The last two options are self explanatory. The Disk menu is comprehensive with an excellent intelligent file handler. Qdesign is able to identify the graphical

file format to be loaded, and automatically loads PD2 or Propublisher or standard QL or Qdesign (compressed/uncompressed) or Atari ST (monochrome) screens into Qdesign. Qdesign can also save in any of the above graphical file formats. The ability to load/save Propublisher screens is important, particularly to *Perfection* (Digital Precision) users who are able to include Propublisher pages into wordprocessing documents. The Disk menu also includes the usual file utilities like Format, Directory, Device, Printer Driver and Configuration file. This menu also allows loading and saving of parts of screens (Cuts) and loading and saving of patterns. Patterns are useful for repetitive effects like carpet design. The Disk menu allows users to load standard QL fonts (or founts) and now Vector fonts (a first for the QL). Vector fonts supposedly give improved on-screen text, analogous to text printed on a laser printer compared with dot matrix printer. Certainly, the vector fonts provided with Qdesign are impressive. However, the pros and cons of one technology over another is outside the scope of this article.

Compatibility

The ability to load standard QL fonts allows compatibility with other graphics programs and font designers. Already three different screen font technologies exist within the QL graphics/DTP market - standard QL, High Definition and Vector. In the interest of compatibility, hopefully software vendors like Digital Precision, J Merz Software, Progs and Dilwyn Jones Computing will modify existing software to cope with all three font technologies. Unfortunately, Qdesign does not include a font editor. A number of standard font editors are available from other software vendors and from PD libraries and J Merz has just released *Vec-Edit*, a vector font designer. To view the fonts loaded, select the Font menu, all fonts are previewed and the user selects the desired font from the list. The ability to perform certain functions/operations on fonts is discussed later. Qdesigns compatibility with virtually all screen/

DTP formats is welcomed.

However, in conjunction with Print menu, the user has a extremely powerful graphics print dump, as powerful as *Grafix* (Digital Precision) but with better user interface. The printer options available include altering the X and Y resolution, print quality (number of print head passes) and Colour (shading on monochrome printers). Additionally, the user can specify Block, Screen, Window or Page. The printer output can be Portrait or Landscape (sideways). Mirror imaging is also supported. The print dump quality is superb, especially with 24-pin printers. Laser output is supported - J Merz Software provides an upgrade version for an extra £7.

A problem with Qdesign was the inability to escape a print dump once started - essential if the scaling was wrong. At this point it is worth noting a fundamental difference between Qdesign and Propublisher. Qdesign sketchpad is white ink on black paper, however, it prints out in reverse, ie black ink on white paper, whereas Propublisher provides black ink on white paper for both screen and printed hardcopy. There are two instances when this becomes important:

(1) Printing a Propublisher screen from Qdesign to take advantage of the excellent printer driver.

(2) Saving a Qdesign screen in Propublisher format for use within Perfection.

In both cases, the DTP page will have to be Recoloured. This function can be performed in both Propublisher and Qdesign. Recolour Window is selected from Options and Recolour part of a screen (known as a cut) is selected from Block. Therefore to recolour a Qdesign page requires using Recolour from two menus. A more satisfactory method would be to have one Recolour selection with the ability to specify the page (unfortunately too many graphics/DTP do not cater for page (global) operations), screen, window or block (cut). Qdesign does allow page colour inversion (why invert instead of recolour?), which should do the trick for normal monochrome print dumps. Other functions available from the Options menu are CLS page/window, mode 8 to 4, reduce, outline, enhance and monoise and swap screen.

Block menu

The Block menu, as its name suggests, deals with cut and paste, essential tools in DTP. However, as above, page cut and paste is not catered for, although this could be easily implemented on Qdesign (select F10, and using Block menu to cut the appropriate section of the page, a magnified window at the bottom of the page just displaying the cursor line would make for a more accurate cut). Block operations are so straightforward that they require very little explanation Cut, Paste, Erase, Move, Recol, Mirror, Rotate and InOut. With Cut and Paste, and Move, the defined block is copied into a buffer and the block is displayed on screen as a boxed outline which can be relocated on the

page using the cursor keys or mouse. Rotate is a combination of 90-degree rotation with the ability to enlarge, however, no box outline is available to indicate the original dimension, which is a pain. Rotate needs to be updated, to conform to Cut and Paste, and to allow full rotation (0 to 360 degrees). A nice clock icon with degree markings would do the trick.

The Status menu is packed with tool functions from brush types to italic slanting (left or right). The Status menu has four functional toggles: Under, Over, Fill and Good - and five new menus - IPWSZ, Blobs, Mode, Csize and VSize menus. IPWSZ is short for Ink, Paper, Width, Side and Zoom menu. Good, which is short for 'precision quality', slows down the following operations: InOut, Mirror, Rotate and Print. As output quality is enhanced, in the same way as for printers, the print speed is sacrificed for NLQ text compared with Draft text. Mode specifies the colour resolution - two- or four-colour - and the graphic resolution. A word of warning: once the colour mode has been selected you cannot change it. Colour mode 4 is the best mode, but requires more memory. Clicking Qdesign or selecting F9 displays the user number and the memory left. It would be nice to see a visual representation of the memory left, at all times. A fuel gauge type icon would be a useful indicator.

The Blobs menu allows the user to select different brush types and previously saved patterns can be loaded into the menu. This is handy for wallpaper or carpet or any other designs that require repetition. Csize and VSize are font tools. Csize is as found in SuperBasic; refer to the QL user manual or QL World's *New User Guide*. VSize is new to SuperBasic users, and offers users many font possibilities, similar to Propublisher toolboxes (Digital Precision). Although the VSize options are powerful, they are self explanatory: Mirror (X and Y), Shadow, Pixel, Bold (Left, Right, Up and Down displaced), IBM/QL font, Csize (X and

Y), Italic (Left or Right slant, angle 0 to 180 degrees), Outline, and Def-Grad. The last option adjusts the angle of a text string on the screen (0 to 180 degrees).

Qsnap

Qdesign comes with an additional utility, Qsnap. Qsnap allows the user to capture either whole or parts of screens from any software. This is of use to game reviewers, or software authors writing documentation or any user needing a screen dump to file. Qsnap was used to obtain the graphics accompanying this article. Once the screen dump or snapshot is on file it can be loaded into any Graphics or DTP program for printing or whatever. Both *Qpac1* or *Qram* (Tony Tebby/Qjump) and *Presgraph* (AFW Software's add-on utility for the communications program Terminal from TF Services) allow screen snapshots, however, Qsnap can also take snapshots of part screens.

The manual is well written, short and concise. It includes information on

Qdesigns file format for those interested boffins, and includes information on incompatibilities. First of all, Qdesign needs expanded ram and disk system and will work with the Atari ST QL emulator, however, the version I tried will not work on QLs with pre-JS roms or Minerva roms below 1.84. At a recent SQLUG meeting, Qdesign was tested on a number of machines with the following observations:

(1) JS + Trumpcard 2 + Qimi: perfect.

(2) JS + Gold card: some of the menu names (Disk, Print, Block, Options) didn't appear on screen at boot up. If you moved the cursor to where they should be and pressed space bar, the menu displayed without further inconvenience.

(3) JM + Trumpcard or JM + Gold card or AH + Gold card: all didn't load.

Graphic 3: Vector Font Options

ESC STATUS	
Mode 4	Zoom 2
Ink 7	Paper 8
Fill	Over
Under	Good
Csize 0,0	VSize
Width 4	Dense 5
Sides 3	Blobs

ESC QK VECTOR-FONTS PARAMETERS			
X - Csize : 10	Y - Csize : 10	Italic X : 90	Italic Y : 90
Def.Angle : 0	Outline : 0	Bold : 0	Shadow Col : 2
Mirror X	Mirror Y	IBM	Shadow Pixel
Right	Down	Left	up
Right	up	Right	up

In actual fact, I was surprised at the number of users still not using JS or

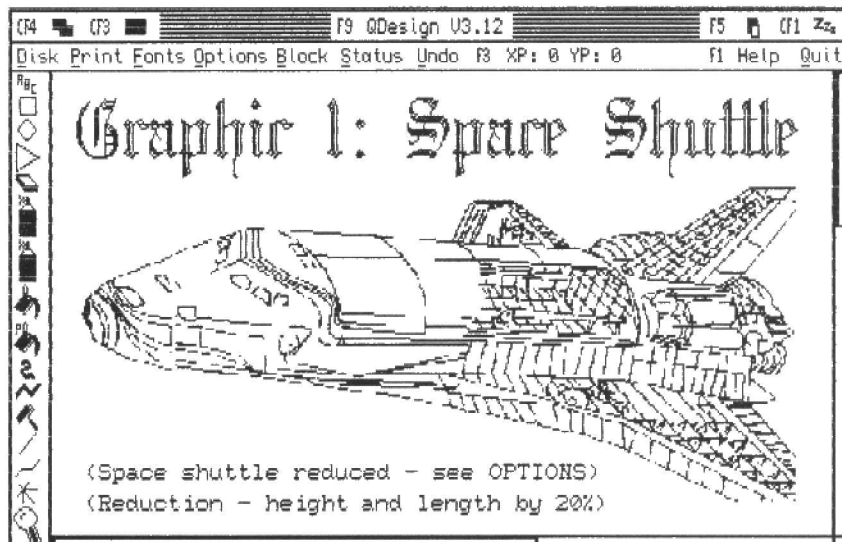
Minervaroms. Overall, I was very impressed with Qdesign. Qdesign will do for graphics that Perfection did for wordprocessing. In addition, Qdesign can double as a printer dump utility and its graphic file compatibility is second to none. The Qpac2 pointer environment certainly enhances the product (remember Qdesign does not require Qpac2 to run as all the necessary extensions are included) and a mouse (via Qimi interface) is essential for the serious user. However, Qdesign lacks the text/wordprocessing file-handling capabilities of PD2 and Propublisher and as such still requires some development if DTP is the ultimate goal. Qdesign will suffer from the JM/AH rom incompatibilities, however, QL users are an ingenious lot and a suitable patch is probably just around the corner. Hopefully, Qdesign will continue to develop, possible milestones being full DTP (V4), 3D-graphics (V5) and animation (V6). Lastly, the ideal companion for Qdesign or for that matter any graphics/DTP program is a hand-held scanner. Any ingenious person who can interface cheap IBM PC hand-held scanners via the QL serial interface is onto a winner.

For further information:

SQLUG: 031 660 1826 (Alan Pemberton - to join)

Care Electronics (authorised Tony Tebby/ Qjump suppliers): 0923 894064

Graphic 2: Sketchpad



A Quantum Leap in QL Wordprocessing

Our new state-of-the-art wordprocessor, **text87plus4** is now available in version 3. **MILES** ahead of any QL application!

text⁸⁷ plus4

+1 USER FRIENDLY TO THE EXTREME

- New well-written manual • Automatic setup and installation • Context-sensitive help • File selector boxes • Commands and key-presses highly compatible with Quill.

+2 THE MOST POWERFUL QL WP

- Extensive navigation and editing • Fast block and search / replace operations • Reformats as you edit • Remembers paragraph formats between sessions • Extensive file commands • Spell Checker with browse and replace • Multi-Window Multi-Document • Page-Preview and Pagination save a lot of time and effort.

+3 UNRIVALLED PRINT QUALITY

- Nothing else can compete in text and character formatting • Fully supports proportional spacing and justifies correctly • Multiple paragraph formats with different margins and line-spacing • Right, centre and decimal tabs • Multiple columns plus headers and footers

- Desktop publishing with several different page layouts in the same document.

+4 FASTEST QL WORDPROCESSOR

Amazing timings on an ordinary QL (much faster with Gold Card) and Atari ST with QL emulator. A 70 page text (24,000 words, 141,000 characters) was used • Load: 25s (STQL 17s) • Save: 37s (STQL 32s) • Search / replace (with auto-reformat) 580 instances: 43s (STQL 14s) • Change Justification: less than 2s (STQL 1s) • Change right margin and reformat 65s (STQL 17s) • Move 10 pages from top to bottom (With manual marking and positioning): 35s (STQL 15s) • Scroll screen 100 lines: 19s (STQL 5s)

Fully compatible with all QL ROMs, Gold Card, ST QL, Thor. Requires disk drive and 256K memory.

Prices (inclusive of Air Mail to overseas)

text87plus4	£ 79.00
2488 drivers for 24pin and Bubblejet printers	£ 19.00
typeset90-deskjet drivers for all HP Deskjets	£ 19.00
typeset90-Epson GQ5000, EPL4100 /7100 lasers	£ 39.00
fountext88 + founted89	£ 39.00

Graphic driver for 9 & 24-pin printers with over 30 founts

Ask for our free, comprehensive leaflets.

SPECIAL OFFER! extended until end of September If you have paid over £59 for any other QL wordprocessor you can upgrade to **text87plus4** for only £59. The manual from the other software is required as proof of purchase (It will be punched and returned). Send cheque, Eurocheque, to: **Software87, 33 Savernake Road, London NW3 2JU**

PERFECTION SPELLCHECKER

Perfection puts a spell on Bryan Davies, without even sexisming him.

INFORMATION

Program: *Perfection Spellchecker*

Price: £49.95 for basic program plus three dictionaries, £29.95 extra for Huge (Mega) dictionary, otherwise enquire supplier. SAE plus disk for small dictionary (existing owners of Spellchecker only).

Supplier: Digital Precision Ltd.

222 The Avenue
Chingford
London E4 9SE.
Tel. 081 527 5493

spellings can be checked, not a list of word-meanings in. This usage leaves the way clear for "word list" to refer to a file containing words not found in a dictionary during document-checking, and stored as a separate, text file, for later conversion into a user dictionary.

One of the prime aims during development was to create really big dictionaries, far beyond that provided by *SpellBound*, *Qtyp* or *Qspell*. Not only that: it was thought worthwhile giving "one in the eye" to PC software by making the biggest dictionary well above the size of anything normally found on PCs. At about 360,000 words, the "Mega" Spellchecker dictionary does not have much competition, on any computer!

Sheer size

Sheer size always creates problems on the QL, which is not the largest of computers in several respects. Memory size and processing speed are at a premium, so Digital Precision have opted to supply several sizes of dictionary, to suit the available ram on users' QLs. The large-

est dictionary (called either Mega or Huge) can be used only when the QL system has 1.5 MB of ram, which effectively means a Gold Card must be fitted, and a HD or ED disk drive. This dictionary file is too big to fit into the Trump Card ram, and too big to fit onto a DD (720 KB) disk. It might be acceptably fast with the basic QL, but the extra urge of the 16 MHz GC processor doesn't come amiss.

The standard big dictionary holds about 220,000 words (the numbers change as development continues, so don't take them as gospel) and can be used on a system with 768 KB Trump Card and DD disk drive. The medium dictionary has 150,000 words; it, also, requires the 768 KB Trump Card. The small dictionary has a mere 77,000 words and can be used with 256 KB Trump Card (or comparable memory expansion) and 360 KB disk drive. Various in-between sizes can be provided, depending upon development and available space on the supplied disks. A word of caution: the memory sizes are nominal, and you may find that you need the next size up for practical use, as you may not find it convenient to have only Spellchecker loaded when checking documents. For example, you may want Perfection loaded, which needs around 140-160 KB for itself, plus the necessary document space (roughly the same space as the file takes up on disk).

On disk

Spellchecker works on the QL, Thors, and the QL emulator on the Atari ST. Most of the dictionaries are too large to be supplied on cartridge; all combinations but the largest are usually supplied on 3.5-inch DD (720 KB) disk, and this one is supplied on HD (1.44 MB) disk. DP has thought about Gold Card owners who do not have a HD or ED drive but wish to use the largest dictionary, and the latter can be supplied "in pieces" on DD, together with a routine to stitch it back together into ram.

The method of checking spelling is unusual; leastways, users familiar with other programs may find the DP approach different. The conventional way is to have the text file (document) loaded in a WP

This program is separate from the *PerfectionWP* program and can be used either in conjunction with it, or on its own. That is, it is not dedicated to Perfection but can be used with text files from other editors and WP programs. It is a more ambitious project than previous QL spelling checker programs, both in the size of its dictionaries and in the facilities it does, or will, offer.

Before going any further, please note that "dictionary" used generally in this context means a list of words against which

```

1992 Aug 05 16:42:16 caps off / memory = 488960
Italic (Off) reformat pPara spell Pages/Block Margins
sUppscript (Off) reformat hEneon Spell as you type Justification
subScript (Off) forced poGe break Reinstate line Tabs
Highlight (Off) back to Normal adL space insert line Wrap
-----
ARLL: Advanced Run-Length-Limited, a method of coding used with hard disc
drives. Similar to RLL, but permitting twice the capacity of MFII (as
against 1.5 times with RLL). Not in common use.

ASCII: short for American Standard for Computer Information Interchange. A
internationally-accepted set of computer representations of 128 normal tex
characters. For example, the character A (capital 'a') is represented in
the computer by the decimal code 65, or the hexadecimal code 41. The user
doesn't normally have to bother about these codes, but they become
important when you try to get what you see on the screen to print the same
way on the printer. Printers often use different codes from those used by
computers. The important thing to remember about ASCII is that text files
-- i.e. those produced by word-processing and note-taking programs -- are
often transferable between programs only in this format. A feature of
converting a normal WP text file into ASCII format is that "special codes"
used for formatting the text -- e.g. typestyle instructions for the printe
-- are removed, so that the text no longer looks quite the same when load
into another program. The major WP programs have file-conversion routines.
-----
MENU 22

```


1992 Aug 05 16:10:41

caps off / memory = 924160

DICTIONARY UTILITY DU v3.02 - MAIN MENU

- 1 .. Extract marked words from, and then unmark, a spellchecked file, and append the extracted words to a wordlist.
- 2 .. Sort a wordlist, removing duplicates, and create a new dictionary from it.
- 3 .. Sort a wordlist, removing duplicates from it.
- 4 .. Configure defaults/dataspace for this program (DU).
- 5 .. Configure file/device defaults for SPELLCHECKER
- 6 .. Quit this program.

Note: No devices should be write-protected.
Press 1 - 6 to proceed.

ex du

program, and to call up the spelling checker, which then takes over from the WP program until checking is completed. As "unknown" (to the dictionary) words are encountered, the user is asked to make a decision about them - typically add them to a supplementary word list, ignore them, or ask for alternative suggestions.

Perfection Spellchecker is called up as a separate program. It takes control from Perfection if that is in use, but it can also be used when no WP (or editor) program is running. From here, though, it diverges. It has three modes of operation - as a concurrent checker of text you are typing into the current Perfection document (but not into any other program); Page or Block mode checking of the current Perfection document; and retrospective checking of existing document files (not necessarily Perfection ones).

Concurrent

The concurrent mode is quite like that of SpellBound when it first came out, which attracted some criticism because it was the only mode available. The retrospective mode is akin to the more useful combination of SpellBound 1 and FileBound, or SpellBound 2. The Page/Block mode is what is usually used on other computers.

The divergence I mentioned may not be clear from the description so far. It arises because, whatever the checking mode, unknown words are not incorporated into existing dictionaries or wordlists during the initial checking process. The unknowns are marked, visibly, and the user has the option to go through the document manually making corrections, or to submit it to processing by the separate Dictionary Utility (DU) program.

The first illustration is of the Perfection screen, with the MENU2> block of options displayed at the top, to show the

spell-checking options; the current document has been checked, but the highlighting of the marked unknown words "ARLL" and "RLL" (twice) is not obvious because of the limitations of the screen dump process.

The second illustration is of the menu options when the Dictionary Utility is started. This latter program has several options; remove the marks from a spell-checked document and add the marked words to a user word list, sort a user word list alphabetically and remove duplicates, sort a word list and remove duplicates then create a user dictionary from it, configure the program itself, and configure Spellchecker. It takes quite a lot of memory (something over 200 KB by default), because it reserves workspace to handle files to be checked, but the reserved space can be user-set when the program is configured.

High speed

With more than one operation needed to complete a spell-checking session, the speed of the individual operations needs to be high to give overall efficiency. There is not much that can be done to speed up the physical side of using the various programs - the keying in of device and file name, selection of options etc. Assuming the user will normally be running Perfection and checking documents after production in it (comment on concurrent checking comes later), the procedure is to EXEC Spellchecker, either prior to or after starting Perfection, use the F3 F3 P (Pages) or F3 F3 B (Block) command to run the marking routine (an interchange of information between Spellchecker and Perfection), then run through the marked document using the Search command to find instances of Strip 2 (the code used to mark the start of unknown words).

The marking is done very fast - 40,000 or more words per minute on a Trump Card QL, several times that with a Gold Card

fitted. This is about four times as fast as the best figures I have seen with other checkers on QL and PC. Strip 2 areas are shown by red highlighting (default colouration) and you have to remove any of these which mark words you do not want added to your word list. You can make these choices when using the DU routine itself, but the error words are then displayed on their own - not in context - and it will not always be obvious whether or not they actually are errors.

One way to remove the Strip 2 code is to find it via a Search, move the cursor one character to the right, then use CTRL+D. CTRL+G restarts the search each time. The aim is to leave only words which are to be added to your word list marked. This can be a lengthy process if there are many marked words which have to be unmarked. Repeated appearances of the same unwanted word do not all have to be unmarked, although Spellchecker marks all occurrences of unknown words, however many times they occur; DU allows you to deal with the first instance of a repeated word then ignore subsequent ones.

EXEC utility

Once the manual checking process has been completed, the file is saved and DU is EXECed. DU gets its job done quite quickly. It can be killed once it is no longer needed, but removing it doesn't appear to increase contiguous memory space.

The initial marking of errors in a file is so fast that it might seem impossible to make it significantly faster, on any machine. Simply marking errors doesn't get the job done, however, and there has to be another, "proofing" stage. You can edit the marked file yourself, and this is all you need do if you have no great problem with your spelling, and no desire to add unknown words to a word list. That is, if you make typing errors, but can normally correct them as soon as they are spotted, there is no real need to continue to the next stage (using the Dictionary Utility).

Having to switch to an external routine and then manually accept or reject the errors naturally adds considerably to the overall time taken. During the review period, there was a fair amount of discussion on speed of WP programs in general, and tests were done on a large (40,000-word) file to see if the overall time taken for a complete spell-checking, -correction and word-list updating operation was materially different in one WP program when compared to another. The Perfection Spellchecker was definitely the faster of the programs being compared. If the file was just marked and then checked manually, the speed difference could be considerable; putting the checked file through the DU routine reduced the performance advantage to maybe 15-30%,

which is still a clear margin. Not making your own word list, to hold any words not in the dictionary used, speeds up checking considerably, but being able to create your own dictionaries is obviously an attraction for many users.

Decisions

Whichever way you go about getting a computer program to check your text, you can't escape having to make decisions yourself. The program says something is wrong, but it is up to you to say what it is. Spellchecker offers no alternative spellings for words it doesn't recognise in the text; if you aren't sure what spellings are, you have to get out a paper dictionary and check. The competitive QL spelling checkers do offer alternative spellings, but the function is very basic. The first few characters of the error word are taken as the basis and a selection of other words having the same starting string is displayed. If the error was of a type where, for instance, the first letter was typed incorrectly - e.g. typing "xar" instead of "car" - any alternatives offered are unlikely to help. DP argue that, unless alternatives can be offered on phonetic and interpretive bases as well as a simple character-comparison one, there is no point in providing the facility; as the QL really hasn't (yet) got the power to handle such comprehensive checking, and the cost of incorporating it could not be recouped in the small QL spellchecker market, it is best left out.

Not all users will agree, as certain types of error are such that comparison-based alternative spellings actually do provide the required answers. Some standard transposition errors, such as typing "thta" instead of "that", will produce a list of alternatives containing the correct spelling, in Qtyp or SpellBound. But you will have to resort to using the paper dictionary at times, whichever QL checker you use.

Checking files outside of Perfection is quite straightforward and requires only Spellchecker itself to be EXECed and the Alt+Capslock key combination pressed. It doesn't appear to matter what the text file type is. Spellchecker goes through and puts a top-hat character (^) against every word not found in the dictionary. Provided the originating WP or editor program can search for that character, it is then easy to find all the unknowns and deal with them. Word lists can be created in the same as way as with Perfection files.

```

1992 Aug 06 15:53:53 caps off / memory = 1176526
F1 HELP          SHIFT F1 last cmd ESC Escape/Cancel  +--+ by char/line
F2 Menu on screen SHIFT F2 Mode8 ++  SHIFT F5 one/two - + SHIFT para/word
F3 Menu 1+2+3+1  SHIFT F3 Menu3+2+1 windows,F5 selects + SHIFT/ALT window
F4 Refresh screen SHIFT F4 Size/Move ALT F5 Over/Insert CTRL C toggle job
Typists may type fast, but magazines scribes should stick to typing real
slow, with no ,,one than tttwo fingers in use at any one too mee.....

The % characters are the markers that Spellchecker puts in at the ends of
unknown words, when working in concurrent mode. You are under no
obligation to do anything about errors at the time -- you can ignore them
and carry on typing, coming back later to fix them. They can be deleted by
the normal CTRL+LEFT/RIGHT keyings. If your QL suffers from a dicky
keyboard membrane, as mine does, Spellchecker's concurrent checking mode
immediately picks up the boobs (in the case of my QL, it does things like
this -- "itttt").

Once the current document has been edited to get rid of mistakes, it can
be run through the Dictionary Utility routine "on auto-pilot", so that all
the unknown words can be added to a word-list.

COL56 LIN15 /16 PRC1 LJ #10

```

User choice

The concurrent checking mode is not something which DP makes a lot of. The instructions indicate that this mode was put in purely because some users will have got used to having it, as owners of SpellBound. DP doesn't recommend it, on the basis that it is obstructive. It certainly does slow down the flow of typing, but QL users in general are not going to be "machine-gun" typists and may not really be unduly disturbed at being stopped every time they type a word incorrectly.

The awful behaviour of the original SpellBound (fixed in SpellBound 2), where it tripped out at the slightest provocation, is a thing of the past. The Perfection Spellchecker behaves well, remaining active even if you move the cursor back to previously-typed text and make changes. It beeps on errors only at the point a word de-limiter is typed, which is usually when you type a Space after the current word. SpellBound beeped immediately a string of characters didn't make sense to it, regardless of whether or not the end of the word had been reached. Having been given the beep, you retrace your steps, make the correction required, then carry on. You don't have to make corrections there and then, as Spellchecker inserts one of DP's collection of unusual characters - a capital X above a small x - after each unknown word and you can edit them later. A small point, but the marker replaces the Space (or other de-limiter) between the unknown word and the next one, and you have to re-type the space after deleting the marker.

An illustration is give here of the Perfection screen during a session of typing with Spellchecker active in the concurrent mode, to show both the marker char-

acters and some of the types of error that are picked up (or not, in one case).

Despite DP's reservations, concurrent checking is eminently usable. It has the distinct advantage that the spell-checking job is over and done with straight away. If you type relatively slowly, don't make mistakes every few words, and make typographical errors rather than being bad at spelling, this mode should be quite satisfactory for you.

Games coming

At the start of this article, it was implied that the current Spellchecker is not the end of the line. A good, big dictionary can be used as the base for a variety of word games. For those interested in word games, this will be one of the main attractions of the program. The intention is to offer a variety of additional functions, making use of the remarkable size of the dictionary and the high speed of checking. Development has been going on for some months, and we should soon see some of the fruits of this.

Those whose memories stretch back to the QSpell checker sold by Eidersoft in the middle 80s will recall that it had some good functions, such as finding all words containing a given string of characters, but was spoilt by slowness and some truly memorable dictionary corruption problems (my copy invented thousands of "words" such as "shortcircuiting" and "sexisming" and added them to the dictionary).

The Perfection Spellchecker will, in future, provide functions for the crossword puzzler, the anagram enthusiast, and anyone else with an interest in words for their own sake. Maybe we will even see some form of contextual and/or grammar checking. The program will then provide both utilitarian and recreational benefits.

DIY TOOLKIT

Simon Goodwin exercises the QL vocabulary and shows how to send neat columns to any Qdos device.

This new *DIY Toolkit* project generates neat formatted reports on the vocabulary of your SuperBasic interpreter. It suits all QL systems from AH to Minerva, and requires only 276 bytes of memory. This means that there's not much hex data to enter, from **Listing two**, to get the versatile VOCAB command on your QL or emulator.

The VOCAB extension was suggested by reader David Moseley of Coventry, prompted by the deficiencies of Toolkit 2's EXTRAS command. "EXTRAS is irritating even for Toolkit 2 on its own", writes David. "With additional keywords being added, and especially with the potential explosion which SET can bring, it is quite exasperating watching them scroll up the screen twenty, or so, at a time."

I have found the same problem when testing new keywords and programs, uncertain of the names known to my system. Even if I set _SCROLL 10 and CSIZE 1,0, to boost the text output speed, the EXTRAS list leaves much to be desired.

Column widths

VOCAB, short for VOCABULARY, has many advantages over EXTRAS. It uses a columnar display, adjusting to the width of the output window or device. If a name is too long to fit the column width it is truncated to ensure at least one space between adjacent columns.

You can configure the column width and default channel to suit your own system, or change the command name with a few POKes, as I shall explain later.

VOCAB shows resident procedures and resident functions separately, so commands and SET functions, among others, appear in distinct reports; VOCAB 9 shows the functions.

These are only two of the ten types of Name Table entry; VOCAB can also report on SuperBasic procedure and function names, arrays, loop identifiers, scalar variables and unset names. The only type beyond its remit is type 1, expression entries which have no corresponding name.

VOCAB takes two optional parameters: a channel number, and a Name

Type. Possible Name Types are listed in the Table, and discussed in most books on SuperBasic and Qdos. VOCAB 0 shows names that would give the value * if PRINTED; these include device names like SER1 and flp1_BOOT, unless they appear in quotes in your program.

Variables

VOCAB 2 shows simple variables, including undimensioned strings, while VOCAB 3 reports the names of DIMed variables, and so on. If you need to document an existing SuperBasic program you can make a good start by loading it and using VOCAB to report the names it uses, in each SuperBasic category.

VOCAB reveals all the names known to the current SuperBasic task, so it is most useful from interpreted Basic and fairly pointless in compiled code. If you use it in a MultiBasic task it shows the 'local' resident names, which may differ from other Basic tasks.

The command needs to know the number of characters that can fit on each line, in order to work out the number of columns available. This is easy enough when the output channel is a window, as the standard Qdos call SD.CHENQ returns the width in characters, taking account of the MODE and CSIZE.

Disappointingly, SD.CHENQ only works if the channel is a window. It is no use for serial output channels such as files, SER and PAR. The QL operating system Qdos keeps no record of the 'width' of these channels. However the SuperBasic system keeps extra variables in its own Channel Table, and these are used if the channel is not a window.

Width default

The SuperBasic WIDTH command sets CH.WIDTH, at offset 34 in the SuperBasic Channel Table. This defaults to 80, which is why the PRINT comma, TO and exclamation-mark separators take a new line after 80 columns. You can change this with the WIDTH command.

This example uses WIDTH and

CHR\$(15) to select a condensed print style, available on most matrix printers. It reports SuperBasic DEF names across 132 columns:

```
OPEN #3,SER1
PRINT #3,CHR$(15)
WIDTH #3,132
VOCAB #3,4
VOCAB #3,5
CLOSE #3
```

It may be that an incomplete line has already been sent to the channel, in which case the VOCAB report might start in the middle of a line, mixed up with the previous output. This is undesirable because it messes up the columns, so VOCAB takes a new line if the print position does not start out at the left margin.

VOCAB always sends a new line when it has finished, like WSTAT, so you can tell that it has finished when no names are found, and the next thing you send to the device does not end up on the same line.

Source code

Listing one is the assembler source code for VOCAB, annotated for study and possible modification. I tested this with Hisoft's Devpac assembler, and the same source should suit other assemblers too. The workings of the assembler code, and possible changes, are explored at the end of this article.

You can obtain the original source code, plus documentation, on disk from DIY Toolkit, at Cwm Gwen Hall, Pencader, Dyfed, Cymru SA39 9HA, UK. Send £7 for DIY Toolkit Volume X (I'm running out of letters!), or call Richard Alexander on 0559 384574 for details of the 20 volumes now available, spanning five years of DIY projects in *QL World*.

The DATA lines in Listing 2 correspond to the code of VOCAB. If you do not own an assembler you can use the SuperBasic in Listing 2 to generate a copy of the code. The hex loader is the same one used for each DIY Toolkit project; you need only type the DATA, from line 590 to line 790 inclusive, if you already have a copy of

the loader.

The loader checks the DATA statements and, if they are correct, generates a small code file which you can load as follows:

```
START=RESPR(276)
LBYTES FLP1
_VOCAB_CODE,START
CALL START
```

Once you have saved the code to FLP1_VOCAB_CODE (or some other file) and entered these commands you can use VOCAB to report the names known to your system.

Configuring

Once the code is loaded to address START you can configure it with a few POKES. These are relative to the start of the code in memory, and take effect before or after you CALL the code. You could use SET, from April 1991, to make the value of START resident and accessible when you have other programs loaded.

POKE START+13,0 if you want VOCAB to list unset names, by default; VOCAB 8 still gives resident procedures. You can POKE any other value from 2 to 8, for scalars, arrays, FOR variables, REPEAT loops, Basic PROCs, FNs, Resident procedures or functions. This POKE just sets the default type listed.

POKE START+11 to set the default output channel. This is currently #1, to suit USE, but you might prefer to divert it to #0, #2 or #3, depending on your set-up.

The most interesting POKE sets the column width; values from 2 to 127 are acceptable, though 2 only lists the first character of each name, with a blank between columns! POKE START+143,10 reserves ten characters per column, suiting the resident commands in Sinclair roms, *Toolkit 2*, *Speedscreen* and *DIY Toolkit* extensions. The default of 16 accommodates *Turbo Toolkit*, *Tascopy*, *Qlib* and other packages used regularly, while allowing reasonable space for SuperBasic program identifiers.

New name

I might have dubbed this command NAMES or WORDS, but decided that these terms were likely to be used already as SuperBasic definition or variable names. If you find VOCAB hard to recall you can POKE a new name of four or five characters into the code after loading:

```
POKE_W START+264,1358
POKE_W START+266,16717
POKE_W START+268,16747
```

Turbo Toolkit simplifies this to POKE start+265,"NAMES". Either way, CALL START to link the new command under the name NAMES.

```
* Sinclair QL World DIY TOOLKIT - VOCAB extension
* Ver. 0.8, Copyright Simon N Goodwin, March 1992
*
start      lea.l      define,a1      Point at name details
           move.w     $110.w,a2      Use the BP.INIT vector
           jmp        (a2)           Link VOCAB to BASIC

*
* VOCAB [ # CHANNEL% , ] [ NAME_TYPE% ]
*
vocab      moveq      #8,d5           Assume type #8 (RES PROC)
           moveq      #1,d7           Assume channel #1
           cmpa.l     a3,a5           Any parameters?
           beq.s      defaults       Is there a # prefix?
           tst.b      1(a3,a6.1)
           bpl.s      no_hash        Flag it for later
           moveq      #-1,d7          CA.GTINT - get integers
no_hash    move.w     $112.w,a2       Return D0 error code
           jsr        bad_exit        Check parameter count
           bne.s      bad_param      Too many parameters?
           subq.w     bhi.s          one_param
           bne.s      one_param      Pick up the name type
pick_chan  move.w     2(a1,a6.1),d5   Pick up the channel #
           bpl.s      0(a1,a6.1),d7   Pick up the channel #
           check_type  It should be positive

*
bad_param  moveq      #-15,d0         BAD PARAMETER error
           bra.s      bad_exit
chan_error moveq      #-6,d0         CHANNEL NOT OPEN error
bad_exit   rts                     All errors exit here

*
one_param  tst.l      d7              Was there a # prefix?
           bmi.s      pick_chan      Parameter is channel #
           move.w     0(a1,a6.1),d5   Parameter is name type
check_type cmp.w      #1,d5          Check for expressions
           beq.s      bad_param      Reject, no name

*
defaults   mulu       #40,d7         Find offset in table
           add.l      48(a6),d7       Add base to offset
           cmp.l      52(a6),d7       Check not beyond end
           bge.s      chan_error      Number is too high
           move.l     0(a6,d7.1),d0    ID for output window
           bmi.s      chan_error      Channel is closed
chan_open  movea.l     d0,a0          Set up Channel ID

*
get_width  movea.l     (a6),a1        Point A1 at BASIC Buffer
           moveq      #-1,d3          Wait indefinitely long
           moveq      #11,d0         SD.CHENQ trap key
           trap       #4             A1 is an A6 offset
           trap       #3             Call Qdos
           tst.l      d0              Z signals ERR.OK
           beq.s      window         Pick up CH.WIDTH
           move.w     34(a6,d7.1),d6   Pick up CH.CHPOS
           move.w     32(a6,d7.1),d0   Pick up CH.CHPOS
           bra.s      get_ready       Are we on a new line?

*
window     move.l     (a6),d0         Find the buffer again
           move.w     0(a6,d0.1),d6   Fetch the window width
           tst.w      4(a6,d0.1)      Is the cursor at left?
           beq.s      got_width      No; move to a new line
           bsr.s      do_newline
           bne.s      bad_exit

*
* D6 is output line width, D5 is name type, A0 is ID
*
got_width  movea.l     24(a6),a3       A3 -> Start of Name Table
           move.l     28(a6),a4       A4 -> End of Name Table
           moveq      #16,d7         Column width (1..127)
           move.w     d6,d4          D4 is room left on line

*
check_name cmp.b      0(a3,a6.1),d5   Check the type of name
           bne.s      next_name      Clear high word of D1
           moveq      #0,d1           Find Name List offset
           move.w     2(a3,a6.1),d1
           bne.s      not_null       Is this a null entry?
           tst.w     0(a3,a6.1)
           beq.s      next_name
not_null   add.l      32(a6),d1        + Name List base address
           move.b     0(a6,d1.1),d0    Get name length (1+)
           cmp.b     d0,d7            It fits the column width
           bhi.s     d7,d0            Emulate Procrustes
           move.b     d7,d0            Allow at least one space
           subq.b     #1,d0
           move.b     d7,d2           D2 is number of spaces
           sub.b     d0,d2            A2 is buffer offset
           move.l     (a6),a2

*

```


Code commentary

Listing one reveals the inner workings of the VOCAB command. If there is only one parameter, a channel number is distinguished from a Name Type by the presence of a hash prefix: VOCAB #0 lists the default type on channel #0, while VOCAB 0 lists unset names to the default channel.

Unfortunately the vectored routine that counts and evaluates the parameters also scrubs the separator details, so we need to look for the hash before calling CA.GTINT. The program sets D7 to a negative value if the hash is found; D5, D7 and A4 are the only registers left intact by CA.GT routines.

The number of parameters winds up in D3; D7 is checked if there is only one, otherwise the supplied values are loaded and checked. Name Type 1 gets 'bad parameter' as expressions have no names. The channel must be open in the SuperBasic Channel Table, or a 'channel not open' error results.

Cursor position

Once the channel ID is known we need to check the width and current cursor position. Display and window channels must be treated differently. The code at GET_WIDTH calls SD.CHENQ to read the cursor position in characters, but this only works if the channel is a window; otherwise the information comes from CH.WIDTH and CH.CHPOS, held in the SuperBasic Channel Table.

Similar code is used in PRINT and INPUT when processing comma, TO and exclamation-mark separators. This explains why the WIDTH command has no effect on display output; it is over-ridden by the window details.

VOCAB moves to a new line unless the cursor is already at the start of a line; SD.CHENQ activates any 'pending newline' which may have been sent to the window but not yet performed.

At GOT_WIDTH we pick up the bounds of the Name Table, which will be searched for the required type of name, in D5. Notice how D1 is cleared before the Name List offset is fetched; this means that unsigned offsets in the range 32768-65535 are correctly processed.

Null entries

Null entries in the Name Table are skipped. This check prevents the first name in the Name List appearing one or more times in the report of unset variables, when VOCAB comes across unused Name Table entries, which consist of eight zero bytes. These redundant entries are sometimes created when the Name Table expands, and cleared out later by NEW.

```
* D0 is length to copy to buffer from 1(A6,D1.L)
*
copy_name  move.b    1(a6,d1.l),0(a2,a6.l)
           addq.l    #1,d1          Advance through Name List
           addq.l    #1,a2          Advance through buffer
           subq.b    #1,d0          Count one less character
           bhi.s     copy_name
pad_spaces move.b    #32,0(a2,a6.l) Add one space to buffer
           addq.l    #1,a2          Advance through buffer
           subq.b    #1,d2          Count one less space
           bhi.s     pad_spaces

*
* There are D7 bytes ready to be output from offset (A6)
*
           sub.w     d7,d4          Will these fit the line?
           bpl.s     they_fit
           bsr.s     do_newline    Take a new line
           bne       bad_exit
           move.w     d6,d4          Start a new line
           sub.w     d7,d4
they_fit   movea.l    (a6),a1       Point at the text
           move.w     d7,d2          Indicate the length
           moveq      #7,d0          IO.SSTRG trap key
           trap       #4            Note A1 is an offset
           trap       #3            Send the string
           tst.l      d0            Did that work?
           bne       bad_exit
next_name  addq.l     #8,a3          Advance down Name Table
           cmpa.l     a3,a4          Have we reached the end?
           bhi.s     check_name    Keep going if A4 > A3
do_newline moveq      #10,d1        Newline ASCII code
           moveq      #5,d0          IO.SBYTE trap key
           trap       #3
           tst.l      d0            Flag the error code
           rts

*
define     dc.w       1            One new PROCedure
           dc.w       vocab-#
           dc.b       5,'VOCAB'
           dc.w       0,0,0        End of PROCs, no FNs
           end

100 REMark Sinclair QL World HEX LOADER v 3
110 REMark by Marcus Jeffery & Simon N Goodwin
120 :
150 CLS: RESTORE : READ space: start=RESPR(space)
160 PRINT "Loading Hex..." : HEX_LOAD start
170 INPUT "Save to file...":f$
180 SBYTES f$,start,byte : STOP
190 :
200 DEFINE FUNCTION DECIMAL(x)
210 RETURN CODE(h$(x))-48-7*(h$(x)>"9")
220 END DEFINE DECIMAL
230 :
240 DEFINE PROCEDURE HEX_LOAD(start)
290 byte = 0 : checksum = 0
300 REPEAT load_hex_digits
310 READ h$
320 IF h$="*" : EXIT load_hex_digits
330 IF LEN(h$) MOD 2
340 PRINT"Odd number of hex digits in: ";h$
350 STOP
360 END IF
370 FOR b = 1 TO LEN(h$) STEP 2
380 hb = DECIMAL(b) : lb = DECIMAL(b+1)
390 IF hb<0 OR hb>15 OR lb<0 OR lb>15
400 PRINT"Illegal hex digit in: ";h$ : STOP
420 END IF
430 POKE start+byte,16*hb+lb
440 checksum = checksum + 16*hb + lb
450 byte = byte + 1
460 END FOR b
470 END REPEAT load_hex_digits
480 READ check
490 IF check <> checksum
500 PRINT"Checksum incorrect. Recheck data.":STOP
520 END IF
530 PRINT"Checksum correct, data entered at: ";start
560 END DEFINE HEX_LOAD
570 :
580 REMark Space requirements for the machine code
590 DATA 276
600 :
```

DIY TOOLKIT

Each name is copied to the start of the SuperBasic Buffer, followed by spaces to fill it to the column width. Long names are cut short to fit. This approach is simple and efficient, although it might seem less elegant than writing the name from the Name List.

There are always at least 128 bytes available in the Buffer, and it's usually quicker to write the whole string from one place, rather than as two parts, name followed by padding spaces. The extra spaces at the end of each line are redundant, but keep the last column tidy if windows overlap.

As each column is output VOCAB counts the number of spaces remaining on the line in D4; if this becomes negative the line is full and the loop calls DO_NEWLINE, before setting D4 to the line-width minus the space needed for the column waiting in the Buffer.

Once all the names have been scanned, VOCAB falls into the DO_NEWLINE routine, which sends the new-line code CHR\$(10) to the device and returns to SuperBasic.

Further options

The code for VOCAB could be extended in several ways. An extra string or unset name parameter might specify a

'wild-card' similar to that used by WSTAT, so that only names which match a certain prefix appear in the report. In some cases it might be desirable to output the names in sorted order, or include other information such as the Name Table index of each identifier.

The defaults presently determined by POKES could be controlled by new commands, VOCAB_COLS, VOCAB_TYPE and so on, which modify the relevant code bytes or data values stored separately. This is tricky if you want the code to work in rom, as the options must be stored at a reliable address in ram.

I am always interested to hear of suggestions or modifications for published DIY Toolkit code, and offer free DIY Vol-

umes for the best suggestions. Please write to me, care of QL World.

VOCAB parameters

- VOCAB 0 Unset Names
- VOCAB 1 Bad Parameter
- VOCAB 2 Scalar variables
- VOCAB 3 Dimensioned variables
- VOCAB 4 SuperBASIC
- PROCedures
- VOCAB 5 SuperBASIC FuNctions
- VOCAB 6 Used REPEAT loops
- VOCAB 7 Used FOR identifiers
- VOCAB 8 Resident Procedures
- VOCAB 9 Resident Functions

```
610 REMark Machine code data
620 DATA "43FA010234780110", "4ED27A087E01BBCB"
630 DATA "67364A33E8016A02", "7EFF347801124E02"
640 DATA "66165543620C6612", "3A31E8023E31E800"
650 DATA "6A1070F1600270FA", "4E754A876BEE3A31"
660 DATA "E800BA7C000167EA", "CEFC0028DEAE0030"
670 DATA "BEAE00346CE02036", "78006BDA20402256"
680 DATA "78FF700B4E444E43", "4A80670A3C367822"
690 DATA "30367820600A2016", "3C3808004A760804"
700 DATA "8704617666B2266E", "0018286E001C7E10"
710 DATA "3806BA33E800665C", "72003233E8026606"
720 DATA "4A73E800674ED2AE", "002010361800BE00"
730 DATA "8204100753001407", "9400245615B61801"
740 DATA "E8005281528A5300", "62F215BC0020E800"
750 DATA "528A530262F49847", "6A0A611E6600FF5A"
760 DATA "3806884722563407", "70074E444E434A80"
770 DATA "6600FF46508BB9CB", "6298720A70054E43"
780 DATA "4A804E750001FF04", "05564F4341420000"
790 DATA "00000000", "*", 22572
```

TF Services

MINERVA

The ULTIMATE operating system upgrade

MI1: MINERVA now with battery for 256 bytes ram, CRASHPROOF clock & interface (I/O) for robots etc. Can autoboot from battery ram.

Other features (common to MI1/MI2):

- Autoboot on reset/ Multiple Basic/ Faster scheduler-graphics (180 of Lightning)-string handling etc/ WHEN ERROR/ Second screen/TRACE
- Foreign keyboard drivers/ "Warm" fast reset etc

MI1 to MI2 UPGRADES: up to 1.79-£5, Rest FREE (send ram (PROTECT LEGS!) and disk or 2 advs)

Original MI1.....£40 MI2.....£65

ALL VERSIONS FULLY GOLD CARD COMPATIBLE

HERMES

A replacement QL co-processor (IC 8049 IPC)

Do you get keyboard bounce?

Do you find fast serial input unreliable?

Do you want to run an MSI 14400 bps modem?

If you can say YES to any one of these, then this is a product you have been waiting for

- Gives up to 19200 bps RELIABLE serial input
- Independent baud rates
- Stops keyboard 'bounce' (unwanted repeat chrs)
- Improves 'fuzzy' and 'random' sound
- Provides extra input and output lines
- Key click
- Reset/INT? invoked more safely

FITTING is a simple job. Simply remove the QL top (8 screws) and replace the chip marked '8049' next to microdrive 1.

PRICE (including manual & software) **£25**

BULLETIN BOARD

OBBS - The first UK QL scrolling bulletin board, now even larger with international ECHOmail and 10mb of files (including the WHOLE King James' bible!)

071-706 2379 U21/23/22/22b1s 24hrs

Qualsoft

Viewdata/VT52/VT100 QL TERMINAL EMULATOR

Multitasking for electronic mail, PRESTEL etc, CET download. Compatible with Tandata Qconnect QL modem/adaptor. Use Qconnect RS232C to drive unbuffered modems. Comprehensive RS manual.

Software (3.5" or ndv) & manual **£30**

FILE TRANSFER - to transfer text AND programs error free QXCONNECT between computers

Available for IBM compats, ATARI ST & Sinclair QL Connects to Psion range with RS232C comes Link Software £7.50 per m/c. Cable £18 (name 2 m/cs)

All prices include VAT and postage in the UK

QL SPARES

Faulty QL boards (excl plug in ICs).....£9

Keyboard membrane...£8 Circuit diags...£2

68000 cpu.....£8 1377 PAL.....£3

3h rom set.....£10 Power supply.....£12

8302 ULA.....£10 8301 ULA.....£10

8049.....£8 HDU ULA.....£12

Other components (sockets etc)....please PHONE

QL REPAIRS

QLs tested with Thorn EMI test rig and ROM software

(HDU hardware extra (if required))

£27 - 6 month guarantee

MAIL ORDER ONLY

12 Bouverie Place, London W2 1RB (tel: 071-724 9853)

SCROLLING BBS FOR ORDERING/QL HELP and FAX - 071-706 2379

★ SINCLAIR QL ★

STOP PRESS

Serial 3 Button Mouse **£45**

Remote Infra Red Mouse **£55**



Externally fitted into SER2, comes complete with interface and software. No internal fitting required, 3 button mouse gives 5 more functions than the QIMI.

A SERIAL MOUSE FOR THE QL AND SER Mouse the Mouse Driver

The SERIAL MOUSE SYSTEM enables a mouse to be used with the QL by simply plugging the mouse into the SERIAL 2 Port, and running a BOOT program. Common to all Mouse Systems, programs with a Pointer Environment are required.

Four Packages are on offer

- 3-BUTTON MOUSE. SER Mouse 3.5 SOFTWARE. D-9pin to QL ADAPTER FOR THE SERIAL PORT. INSTRUCTIONS. **£45**
- *1A. AS FOR 1 ABOVE, but including PC MOUSE DRIVER DISK. **£12 extra**
- CORDLESS (INFRA-RED) 3-BUTTON MOUSE. SER Mouse SOFTWARE. D-25pin to QL ADAPTER FOR THE SERIAL PORT. BATTERIES & INSTRUCTIONS **£55**
- *2A. AS FOR 2 ABOVE, but including PC MOUSE DRIVER DISK, and 'VISUALIZER' a PC Mouse Painter Program on 3.5 and 5.25 disks. **£18 extra**

* While stocks last.

NB. We still have lots of QL's, 1 & 2 Mb disk drives, peripherals and components. See previous adverts or contact us for details

EEC LTD

18-21 MISBOURNE HOUSE,
CHILTERN HILL,
CHALFONT ST PETER,
BUCKS.
SL9 9UE

Tel: 0753 888866

Payment terms:

CWO, Access, VISA et cetera

Cheques - allow 10 days

Delivery:

Postage - £3

Product subject to availability. E&OE.

Fax: 0753 887149

ONE MAN'S SYSTEM

A stroke of genius? David McCullagh has trained *Archive* to keep his competition team (and its accounts) going swimmingly.

It now seems a lifetime ago when I first took delivery of my QL. I was locked away for weeks coming to terms with *Quill* and *Abacus*. *Archive* seemed beyond me. Initially my main uses were report and letter writing and basic accounts. The furthest I got with SuperBasic was writing boot programs.

My Medic experiences are long gone and my setup now includes a JS QL with an 896K Trump Card (soon to be replaced with a Gold Card), *Lightning*, a Care regulator, Miracle hard disk and Miracle 3.5 in drives, a Keyboard 90 IF, a PC-type keyboard, a Philips Green Screen, recently an EEC Sinclair Colour Monitor, and a Star XB24-10 printer. I had intended to put all the QL parts in a Spem box but with the arrival of the hard disk, I will have to look at a PC-type housing. I also have a twin-floppy Thor 8. It was in constant use, but its limit of 640 K and its inability to take any extra hardware now makes it too restrictive. Aside from various spare parts which will someday form a backup QL, this is my set up at home.

At work I have a twin-floppy Thor XVI with 1 MB, a Philips Green Screen and a Star NL 10 with sheetfeeder. All machines use a Miracle Centronics interface. Boards, roms, power supplies, etc. have been replaced on all machines over the years, but my soldering has not improved! I have tried to network the machines as per QL World and Quanta articles but without success. I will try again, so that I can use a backup QL as a print buffer for the many long printouts I make.

Demise and reprise

I had thought that my ideal machine was a Thor XVI with 1MB, and a 40 MB hard disk. Regrettably with Dansoft's effective demise this is not very practical. The Gold Card and the plans for a Super QL seem the only realistic option. So the future still looks bright.

My software experiences give truth to the adage: 'You use 10 per cent of your software, 90 per cent of the time'. My main programs now are Cowo's *Qtop*, Psion

Xchange, *Archdev*, *Abacus*, *Perfection* and *The Editor*. *Lightning* and *Toolkit II* are of course installed. My wishes are for a good but flexible accounts package, and for *Xchange* on the QL. I have *Cash Trader*, but could not get it to do exactly as I wished. A good *Abacus* spreadsheet covers wages.

My main job is as manager of a sports centre in Dublin, while my second, unpaid, job is as manager/coach to a national level swimming club. In the former role I use the QL/Thor to write letters, reports, and so on; I write *Archdev* programs to maintain bookings/invoicing information and to maintain records, produce invoices and membership listings, and that sort of thing.

The first of the main database programs I use is the Bookings Module, which takes information from groups who hire our facility, calculates term lengths and the costs involved, and produces invoices. I can produce lists of groups who start or end in a certain month, who owe money, when they paid, etc. I store the start and end dates in two forms. The first is dd/mm/yy, and as this is difficult to order, I also store it numerically by adding the 'dd' to 'mm' x 31 to 'yy' x 365, giving a number such as 726913. This permits the file to be ordered on start or end date and, by using the INT function, I can calculate the number of days in a term.

Books and accounts

Three database files are used for 'bookings'. The first 'd' contains the details of the groups, such as names and addresses. When a group starts a new term, this information is taken from 'd' and passed to 'b' where the actual term details are added. A third file 'c' contains just one record, the current invoice number.

The next database is a very basic accounts program. We use a cash register which produces daily readings. Our auditor requires a page per reading, so the readings are entered into a computerised Day Sheet which simply stores the readings for Income and Expenditure and gives a daily balance. These are then printed out each month. A second file is maintained which, when the

last day of each month is reached, creates a Month Totals File. This file will then contain 12 or maybe 15 records, if I want a six and twelve month total.

I plan to expand this to accept cheque details and lodgements. The last stage would be to combine this information to produce a monthly Income and Expenditure Sheet with current balances. My knowledge of accounts is very limited, so producing an Audit is beyond me and not really useful to me.

Stock control, etc., are not big issues and you have to be careful not to waste time on a computer if a manual system is quicker and functional.

In my work as manager/coach, I find even more use for a computer. The club has over 100 members and, being a sport based on timing, their recordings are essential. The first of three databases holds all current best times of each swimmer, the Personal Best Module. The file is constructed so that each record contains all of a swimmer's events and times. I could reprogram to use a record for every event and time for each swimmer, but aside from the great size of such a database, it is limiting, without a lot of programming, to produce one sheet with each swimmer's event profile.

Controlled fields

There is also the point that there are a set number of events in swimming, therefore the number of fields in the database is controlled. Again the format of menus is used with a 'sedited' screen doing much of the work.

The second 'club' database is the Clubil Module. Swimmers pay monthly and annual fees, and also take part in competitions for which entry fees are chargeable. Here I chose a different approach, in that every entry becomes a record. This leads to a large database, almost 5000 files, but it allows more manipulation of the data, and as the entries per swimmer increase there is no danger of running out of fields. A details file ('d') is kept, holding names, etc., and the swimmers' account balance. Every time an


```

Proc bilbac
REM Called by Shutdown to save .dbf from Ram to Flp or Win
SHOWHEAD : local cont$,an$ : print at 5,5;un$;"Save Files...";un$
while not instr("YyNn",cont$) : input at 9,5;"Continue (y/n) ";cont$
if instr("Yy",cont$) : kill fn1$ : kill fn3$ : kill fn5$ : print at 11,5;"Now saving..."
backup fn2$ as fn1$ : backup fn4$ as fn3$ : backup fn6$ as fn5$ : endif
if instr("Nn",cont$) : endif : endwhile : while not instr("AaRrQq",an$)
print at 15,5;"Another copy/Restart/Quit (a/r/q)? ";an$
if instr("Aa",an$) : SELDEV : BILBAC : else : if instr("Rr",an$) : START1
else : if instr("Qq",an$) : kill fn2$ : kill fn4$ : kill fn6$
QUIT : endif : endif : endif : endwhile
endproc

Proc mmenu
REM Calls Procs after input from Showmenu. Set "mm<" to last option +1
while mm<11 : if mm=0 : SHOWMENU : else : if mm=1 : INMET : FINDS : ENTRY
else : REM repeat "mm=" for other ShowMenu options
if mm=10 : SHUTDOWN : endif : endif : endif : SHOWMENU : endwhile
endproc

Proc orcb
REM Called by START1, opens Ram1_files
open fn6$ logical "c" : open fn4$ logical "d" : open fn2$ logical "b" : sload fn8$
endproc

Proc seldev
REM Select Data Device, choice of Flp1_, Flp2_ or Win1_
SHOWHEAD : let dv$="" : let dev$="" : print at 5,10;un$;"SELECT DATA DEVICE";un$;dv$
print at 10,10;"A = Flp1_" : print at 11,10;"B = Flp2_"
print at 12,10;"C = Win1_" : while not instr("AaBbCc",dv$)
input at 20,10;"Select Device (a/b/c) ";dv$
if instr("Aa",dv$) : let dev$="flp1_" : else : if instr("Bb",dv$)
let dev$="flp2_" : else : if instr("Cc",dv$) : let dev$="win1_"
endif : endif : endif : endwhile
endproc

Proc showhead
REM Screen Message
cls : print ;"TEMPLEOGUE SWIM CLUB - Invoice Module"; tab 60;"Date ";today$
print tab 60;"Time ";time() : print tab 60;"Kbytes ";(memory())/1000
endproc

```

entry in made, the 'd' data is appended to the 'i' (Invoice) file, the entry details are made, and the 'd' file account balance is updated. An invoice is then issued to each swimmer monthly.

The third Module is used by both the previous two. The Details Module holds names, addresses, phone numbers, etc. In itself, it is simple in that it is a flat file with one file per swimmer added or deleted. However, it is central to the other two databases.

Sorting swimmers

A future project is to rewrite the Personal Best Module along the lines mentioned, of one file for each swimmer's event. The reason for this is that many competitions have qualifying times. With a file per event, it would then be possible to call on all swimmers of a certain age who are inside a certain qualifying time (stored in a further database). It would also make ranking the swimmers easier, such as fastest to slowest in an event.

I use Archdev because the Archive system of Xchange is, in fact, Archdev. I also use it because I sometimes use

ArchRtm or Run-Time Archive. The advantage of ArchRtm is that it is faster than Archive/Archdev in loading, etc. The disadvantage is that Esc (unless it is trapped), causes the program to quit. It was meant to be used by commercial suppliers, as it tokenises code and users cannot access the programs. You must write _prgs in Archdev to use them in ArchRtm, so it's as handy to use Archdev all the time, although I notice no advantage using it over Archive.

Archive is a bit daunting at first but I found that with help from Chas Dillon through his Quanta Notes booklet (which aside from being ideal to learn how to plan a large program gives many useful tips and information not available elsewhere, on Archive), Mike O'Reilly's *Database Management on the Sinclair QL*, Albert Russell's *Managing data on QL Archive* and Malcolm Johnston *Advanced Archive on your Sinclair QL*, that I can now write passable programs. I know that my coding is long winded (some listings reach 600 lines) and with more knowledge it could be more efficient, but I find that it works. In fact, the main problem is Archive's lack of speed in selecting, searching, etc. I have

looked at *Flashback* and others, but Archive wins for me because of the limitless tasks that its programming language can help achieve. My coding may also be long winded because I never finish a program! Well, I do, but then I start fiddling and see additions that I can make and improvements in coding, sometimes years later.

Helpers all

I use the same basic system to write all my programs. I 'borrowed' it from Chas Dillon. The listings I have included indicate the skeleton _prgs I use. I start up with something like "Run Object 'flp 1'", where Clubil is a _prg file saved with 'object' to become _pro file, making for faster loading. 'Run' requires a procedure called Start. Here I set up variables, then pass control to Seldev which gives a value to dev\$, so that I can load data files from floppy or hard disk. It also gives a value to fn1\$, etc., which are the global filenames. The files are backed up to ram, opened using ORCB ('open ram club bill') and control is passed to Showmenu, which is the programs 'front end' or menu from which non-printing

options are selected.

I have included a few options in the listing, but you can have as many as you wish. On selecting an option, control is passed to MMenu, where the option selected is passed to the procedure controlling that option. I have not listed the details of these, as they are specific to each application. However, the most common ones reset all variables in a record, call a screen (sedit), ask for input using Sinput, then if you confirm that the information is correct, the record is appended to the database.

Selecting option 9 calls up a submenu which operates like Showmenu but which controls all output to the printer. In Clubil, I can print out all due invoices, select one member and print out a due invoice, print out all account balances etc. When printing is completed, control is returned to Showmenu.

Selecting 10 from here leads to Shutdown. This procedure closes all files, then calls Seldev to select the device to which Bilbac will backup the ram based data files. Normally it's still the same device, but I always keep two copies of data files so I can Bilbac to flp 1_ and when completed I can Bilbac to flp 2_, thus saving disk changes. Also as I have a hard disk at home and none at work I don't have to switch to SuperBasic and WCopy RAM1_dbf, WIN1_dbf, Seldev offers me the option to

copy to WIN1_. On completing the backup, I can return to Showmenu or Quit. Although I have not had lockups or crashes in a long time, thanks to Care's power regulator, I still like to backup every 15 or 20 minutes. Good habits save a lot of tears.

Using this system, I never leave a session without choosing Option 10, and so never have problems forgetting to close files. In fact, I have had no file corruption in a very long time, in part because all work is done in ram (which is the fastest way of manipulating data in Archive), and in part because of option 10, and also, I think, because I don't use microdrives.

I am weak on using parameters in Archive procedures. I know they reduce coding and are faster, but I never got the hang of them. Soon I will take a week off, rewrite the programs and use parameters - when the rush is over!

I have been told that PC-type databases such as DBase require good programming skills and just as much frustration as Archive. Archive is limited in that it is difficult to get to grips with initially, and it is relative slow but overall I have not yet found any serious job which it could not handle. Compared with today's databases, it does not have graphics, menus, etc., but it is free. I see that PC Archive will cost nearly £300, and I have not heard any plans to convert it for use on the QL.

I have always thought that the main

uses of computers were for word processing and database activities, and I find it surprising that no one has made a serious attempt at producing a better Archive/DBase-type of database for the QL.

PC not for me

I had considered moving to the PC world, but experience with one, the investment I have in the QL, and the very high cost of PC software put me off. With the arrival of the Super QL, either through putting it together yourself or buying one off the shelf, the QL looks stronger than ever. The other evening I stopped, struck by the versatility of switching from The Editor to SuperBasic to Perfection to Archdev, etc., all in 1 MB. I know PCs can at last do similar, but not as easily nor as cheaply.

Being in Ireland where there is no QL supplier, and where anything you can get is very expensive, and I am indebted to people such as Chas Dillon, Freddy Vachha and Digital Precision, *QL World*, especially Simon Goodwin, Bryan Davies, Miracle Systems and many Quanta members without whom I would never have gotten this far. My thanks to you all.

If any readers want listings of any of my programs I will be happy to supply them, but there are few rem statements and deciphering them may be difficult.

SUBSCRIBE TO SINCLAIR QL WORLD!

SUBSCRIPTION FORM

Rates (12 months)

U.K. **£23.40**
Europe: **£32.90**
Rest of World **£40.90**

Select back issues available on request. Please call for details.

Cheques should be made payable to Arcwind Ltd.

Please debit my credit card/Access/Visa

Card No. _____

Expiry Date _____

Please send me a 12 Month subscription to Sinclair QL World

Name _____

Address _____

Telephone No. _____

Signed _____

Date _____

Send the completed form to:

Arcwind Ltd
The Blue Barn,
Tew Lane
Wootton
Woodstock
Oxon OX7 1HA
Tel: 0993 811181
Fax: 0993 811481

ARCHIVE

A = N + S - W = E + R - S

I Robin Stevenson brings to book a bibliography routine - but it has many other uses as well.

If you have ever written some serious non-fiction, the chances are you stuck a bibliography at the end (*nicely put, Robin*). Whether it's a school or college essay, a four-page village history, or a full-blown book, you should provide one, both to list your sources, and guide the reader for further information.

If you are already a database fan, then you won't need me to tell you that the best way to compile your bibliography is to computerise it. All the classic database elements are there: a mass of disparate, but structured information (authors, titles, etc.) to be placed in alphabetical order, and presenting in an orderly manner. The first part of that process is a simple *Archive* problem. Identify the various fields you will need, insert the data, and use the 'order' command to put it in order.

Extraction

As with most databases, the tricky bit is extracting the information in the way you want it. If the bibliography is just for books, the situation is fairly simple. You print out the author, title, publisher, and date, with suitable punctuation inserted. But there are complications. The reference may be to a chapter in a book edited by someone else. So the editor, and the overall title need to be included. It may be an article in a journal, in which case the journal name and issue number should be included. There are other formats for newspapers, legal documents, etc., but we shall restrict ourselves to these three. You could add other specialist headings

BIBLIOG_PRG - Listing for Bibliography Database Management.

```

proc NewFile;File$
  create File$ logical "bibliog"
  Author$
  Title$
  Year
  Publisher$
  City$
  Compilation$
  Editor$
  Journal$
  Issue$
  Comment$
  UseThis
  endcreate
endproc

proc Identify
  reset : order Author$;a,year;a: display
  print at 13,12;"Include this Record? Y/N : ";
  while not eof()
    sprint : print at 13,40;" "; at 13,40;
    let answer$=upper(getkey())
    if answer$<>"Y": let answer$="N": endif
    print answer$
    let UseThis=(answer$="Y")
    let bugfix=recnum(): update : position bugfix
  next : endwhile
endproc

proc IncludeAll
  reset : first : while not eof()
    let UseThis=1
    let BugFix=recnum(): update : position BugFix
  next : endwhile
endproc

proc OutputTo;X$
  if upper(X$)="P": spooloff
    let Ucode$=chr(16)
    let LeftMargin=10
    let WrapMargin=18
    let RightMargin=70
    return
  endif
  if upper(X$)="S": spoolon screen : cls
    let Ucode$=chr(5)
    let LeftMargin=3
    let RightMargin=61
    let WrapMargin=11
    return
  endif
  spoolon X$ export
  let Ucode$=""
  let LeftMargin=0
  let RightMargin=80
  let WrapMargin=0
endproc

proc PrintInit
  let CurrentMargin=LeftMargin
  let CurrentPos=LeftMargin
  let Underlining=0
endproc

proc PrintLine;X$
  local Pos,Max,H$,Sep$,Uline$
  let Sep$="- ,.:;""!?"
  if Underlining: let Uline$=Ucode$
  else : let Uline$="": endif
  let Max=RightMargin-CurrentPos
  while len(X$)>Max
    let Pos=Max: if pos>1
      while not instr(Sep$,X$(Pos)) and Pos>Max-10 and Pos>1
        let Pos=Pos-1
      endwhile : endif
  endwhile

```


yourself if required.

Finally each entry needs to be correctly formatted. The name needs to be 'outdented' from the rest of the text, and lines need to be wrapped around properly. *Quill* does all this without a murmur, but Archive needs a little persuasion - in this case, via a slightly tortuous, but wholly reusable Archive procedure. There are also procedures to create a file, and extract the bibliography details. As a little embellishment, it allows you to select only part of the list before printing it out. You can build up a comprehensive reference on a subject, and only include the entries you require for a particular publication.

Your complete bibliography could then be a valuable research aid, using the normal Archive locate, search and find facilities. There is plenty of scope for adding to these features. You could add a field to record where the book is kept ('shelf 3b', 'loft', 'reference library', etc.). And if books are not your

thing, it would be very easy to adapt the program to organise your record collection, photographic slides, or whatever, with fairly minimal changes to the program.

Once the listings are entered, debugged and saved to disk/microdrive, you can start to enter your data. Rather than providing a complete program shell, we shall use the Archive 'Command Line Interface'. Don't let that phrase panic you. It just means typing commands at the normal Archive prompt. Some of these will be standard Archive commands (such as Insert, or Close). Others are the program commands such as NewFile or PrintList. In case you forget the commands, there is a 'help' command, which will describe the commands (both Archive and Bibliography) you may need.

Using it

Once you have tried it, the program should seem quite simple. Here is a brief description of how to use it. To create a new data file, you enter:

NewFile;'filename'

choosing a filename that describes its contents. When you return to a previously created file, you can use the standard Archive command Open 'filename' instead. You can now enter details of the books you wish to include, using the ubiquitous 'Insert' command. Obviously you can keep returning to a file to add extra books, as your research/enthusiasm grows. For each entry you will need to decide which of the three entry types it is, as the different types use different data fields. All types should include the author (or authors all on the same line), the title of the piece, and year of publication. If it is a journal/magazine article you should leave the next few fields blank, and add the journal name, and details of the issue and page numbers.

```

if Pos<=1: lprint : else
  if not instr(Sep$,X$(Pos)) and len(X$)>Pos+7
    let Pos=Pos+7: let H$="-"
  else : let H$=""
  endif
  lprint tab CurrentMargin;Uline$;X$(1 to Pos);H$;Uline$;
  let CurrentPos=CurrentPos+Pos
  let X$=X$(Pos+1 to )
  while X$(1)=" " and len(X$)>1: let X$=X$(2 to ): endwhile
  if X$=" ": let X$="": endif
  if len(X$): lprint : endif
  endif
  let CurrentMargin=WrapMargin
  let Max=RightMargin-CurrentMargin
  if len(X$): let CurrentPos=CurrentMargin: endif
  endwhile
  lprint tab CurrentMargin;Uline$;X$;Uline$;
  let CurrentPos=CurrentPos+len(X$)
endproc

proc Book
let Underlining=1:PrintLine;Title$: let Underlining=0
PrintLine;" ":PrintLine;Publisher$+" ":PrintLine;"("+City$+" )"
endproc

proc Journal
PrintLine;Title$+" ".
let Underlining=1:PrintLine;Journal$: let Underlining=0
PrintLine;" ":PrintLine;Issue$+" ".
endproc

proc Compilation
PrintLine;Title$+" ".
PrintLine;"In "+Editor$+" , editor, "
let Underlining=1:PrintLine;Compilation$: let Underlining=0
PrintLine;" ":PrintLine;Publisher$+" ":PrintLine;"("+City$+" )"
endproc

proc PrintEntry
PrintInit: lprint :PrintLine;Author$+" ", "
if Editor$=""
  if Journal$=""
    Book
  else :Journal
  endif
else :Compilation

```

```

endif
PrintLine;str(Year,2,0)+" ". :PrintLine;Comment$: lprint
endproc

proc PrintList
local Spool$
print at 13,0:"Send To [P]rinter [S]creen or enter filename: ";
input Spool$;OutputTo:Spool$
order Author$a,Year;a
select UseThis>0
first : while not eof()
  PrintEntry
  next : endwhile
spooloff
endproc

proc Help
cls : print
print "
print "
print " To Create a new file, NewFile;<filename>"
print " To Open an existing file, Open <filename>"
print " To Insert new records, Insert"
print " To Identify the required records Identify"
print " To Include the entire database IncludeAll"
print " To Print the bibliography PrintList"
print " To Close the file after use Close"
print
print " To modify margins, Edit the procedure 'OutputTo'"
print " To modify underline codes, Edit procedure 'OutputTo'"
endproc

```

If it is a book, the journal information is left blank, but you should add the publishers name, and city of publication. And if it is only a chapter in a book compiled and edited by someone else, you should include, in addition to the publisher information, the editor's name(s) and the overall title of the compilation.

Fairly crude

The printing routine uses the fairly crude technique of seeing which fields are blank to decide which type of entry it is. So make sure you don't accidentally add spaces or other characters to fields that should be blank. Punctuation between entries is added automatically, so you should not duplicate this. The exception is within names of authors and editors. Names should be entered surname first, followed by a comma, and then initials, followed if desired by a full stop.

Finally, there is an optional 'Comment' field. This is particularly useful if the bibliography is for 'suggested further reading'. You can use this to point out particularly important books on a subject.

If you need to correct or alter entries after you have entered them, you can use the normal 'navigation' commands ('first', 'next', 'find', etc.) to get at the record, and use the 'Alter' command to make the changes.

If you find a single line is too short to fit some entries on, the best solution is to design your own input screen, using sedit. You can drag the dotted field area down to cover two lines, in the fields that require more space. You will then need to include appropriate 'sload' and 'screen' commands to ensure your custom screen is available.

Selection

When you come to make a print-out, you need to decide whether you want all of the entries, or just some of them. If the former, you should first enter the instruction:

IncludeAll

which will mark all the records as being required. If not, you should enter:

Identify

This steps through all the records in turn, asking if they are to be included. An answer of 'Y' will set the field 'UseThis' to 1, whereas 'N' will set it to 0. Only those set to 1 will be included in the printed list.

The printing instruction itself is the command:

PrintListYou

are given the option of sending the output

HELP	ALTER/INSERT	COMMANDS
press F1	ENTER or TAB to go to the next field	press F3
PROMPTS	SHIFT/TAB to go back to a field	ESCAPE
press F2	F5-enter record as displayed F4-exit	press ESC

Logical name : biblio
 Author\$: Spoehr, A.
 Title\$: Cultural Differences in Natural Resource Uses
 Year : 1956
 Publisher\$:
 City\$: Chicago
 Compilation\$: Man's Role in changing the face of the earth
 Editor\$: Thomas, W.L.
 Journal\$:
 Issue\$:
 Comment\$:
 UseThis : 1

to the screen (by entering an 'S'); to the printer (by entering 'P'); or to an export file, for which you should supply a filename. Such a file could be imported into Quill or another word processor for further editing. The various margins and underline codes are all set differently for each of these three output options. To change them you simply edit the procedure 'OutputTo' before calling PrintList.

The program will underline the publication title, which will be different for each of the three publication types. By sending CHR(16), the printer driver underline instruction is toggled on and off - whereas on the screen, CHR(5) does the same job.

Hand-painted

Unfortunately, Quill does not seem to import the underlining instructions, so you would have to go through and 'paint' them in by hand. If anyone has found a way round this, *Open Channel* would just love to hear from you.

Finally it may be useful to look briefly at the text formatting procedure 'Printline' shown in the listings. It is a recurrent problem when printing from Archive, that lines may need wrapping around, or otherwise chopping up over several lines. The procedures OutputTo, PrintInit, and PrintLine are all required for this, and could be used in any Archive application.

PrintLine uses global variables to keep track of required margins, and current position on the line. This means you can keep sending as many pieces of text to PrintLine as you like, and it will continue adding to the line it started previously, and wrap around correctly at the end of the line. It will only start a new paragraph when you re-initialise the posi-

tion variables, and lprint one or more blank lines.

Apologies

As I said earlier, it is a tortuous and more-or-less unreadable bit of programming, for which I apologise. Its convolutions are due to the number of exceptions it has to catch. The best that can be said about it is that it works, which ultimately is what matters.

The way to use it is as follows. Before printing can commence you should call OutputTo, to set the margin and underline variables. This will also redirect output to the chosen device. Before each paragraph you should call PrintInit, to initialise the current margins and position. From then on, you can keep calling PrintLine with as many text items as you require. After the first wrap-around the 'WrapMargin' value will be used for the left hand side. If you set the variable 'Underlining' to 1, text will be underlined until the end of the paragraph, or until you reset it to 0.

At the end of the paragraph, you must, in addition to calling PrintInit, send at least one lprint, to move on to the next line. If you want a gap between paragraphs, further lprints are necessary.

By allowing any number of separate PrintLine calls within a paragraph, there is no limit on paragraph size. If you gather all the text into one variable before formatting it, you are stuck with Archive's 255-character limit. The approach used here makes a more complicated program, but allows for much more flexible database presentation.

If you have any Archive problems you would like to see solved on these pages, please write and tell me about them, care of *Sinclair QL World*.

Systematic Machine Code Programming

In the last in the current series, Alan Bridewell explains how to combine machine code routines with SuperBasic programming for greater efficiency

So far in this series, I have only considered the subject of machine code programming in terms of writing complete multi-tasking programs, entirely in machine code. All the examples given have been of this type. There are many advantages to this type of programming. The code is about as fast and compact as you can make it, and the most flexible. If it is possible to do something on your QL then it must be possible to do it with machine code. Most other programming languages will impose some constraints. Even if they theoretically allow something, often the practical limitations of speed and size of program make it unworkable.

The big disadvantage is that the assembler language we use to produce machine code is by no means easy. This whole series has been about trying to make it easier, and I hope I have shown that with a bit of effort it is possible to put together useful programs without getting completely bogged down in the minute details of the language, provided we start from chunks of code which do recognisable jobs, rather than the single program lines which make up the language. But even doing things this way, it is still a very daunting prospect to write a major program entirely in machine code.

Own library

I have quite a library of small machine code programs I have produced for my own use. Some are very useful; others are interesting rather than useful. Most are what I would describe as experimental, in other words, just trying out ideas. But I have to confess that I have never produced a large program entirely in machine code, and perhaps I never shall.

Whenever I try a major project, I fall back to one of the high-level languages, usually either Forth, or SuperBasic, which I then compile. The reason is simple enough. These languages are so much easier to follow and program. This, of course, is the reason they were invented.

You may be getting the impression that I am trying to tell you that when it comes to major projects, machine code is alright for the whiz-kids writing for the leading software houses, but not for mere mortals. Well, that's not what I am saying. For a start, if you have a talent for programming (and you're twenty years younger than me!) then you ought to have a go. You never know what you can do until you try.

The message

However, the real message is this: you do not have to write complete programs in machine code. You can write short routines which can then be incorporated into larger programs written in other languages. This can make the program faster and more concise than if it had been written entirely in a high level language. It can also make the language more flexible, and enable you to do things you may not have been able to do if you had stuck purely to the high level language.

For most people, the only high level language they will be familiar with on the QL is SuperBasic. So for the rest of this article I shall confine myself to using machine code with SuperBasic. There are two ways of incorporating your machine code into SuperBasic.

A machine code routine may be loaded into the QL ram by first allocating some ram space for it using the Respr function, and then loading it into the space using the Lbytes command. For example, suppose you wished to load a routine called Fred from a disk in flp1_, and suppose "FRED is

500 bytes long. Then we could start our SuperBasic program with the line:

```
z = RESPR(500): LBYTES flp1_FRED,z
```

and the routine would load into ram, starting at address z. If at some later part of the program, we use the line

```
CALL z
```

the program will start to execute the machine code routine from z. When the machine code has completed, it will return to the next line in the SuperBasic program, as long as the machine code ends with the two lines:

```
MOVEQ #0,D0 ; RETURN NO ERROR  
CODE  
RTS ; RETURN FROM SUBROUTINE
```

Without the first of these lines, the machine code will return with an error message and stop the SuperBasic program. Without the last one the routine will not return to SuperBasic, and the QL will probably crash!

There is a further important feature of the Call command in SuperBasic. It can take up to 13 additional parameters, which will load in order into the data registers D1 to D7, and then the address registers A0 to A5 before executing the machine code. This means that the same machine code routine can be made to do different jobs in different parts of the SuperBasic program, by starting with different parameters. This can clearly lead to a great deal of flexibility in use.

There is, as always, a down side to all this. Up to this point in the series, I have paid little attention to error trapping. Error trapping means that if some data goes into the program so that it cannot execute properly, it does not crash, but comes up

with an error message, and still leaves us (or the program) in control. The reason I have been able to ignore it is because if all the data comes from within the code, there should be no errors unless there is a mistake in the code. The only occasion in the series where this was not the case was the file transfer routine in part 2. In that program, the user had to key in device names for input and output channels. Without error trapping, any typing errors there would have crashed the program.

Outside errors

Clearly, if the SuperBasic program can change parameters used by our machine code, there is room for errors to arise from outside the machine code. So we must carefully trap them to make sure they do not crash the QL. In most cases, this is a very simple process. The majority of Qdos trap calls and vectored routines leave a zero in register D0 if there is no error, and leaves a negative number if there is an error. The value of the negative number indicates the type of error. So after the call is completed we use the lines:-

```
TST.L D0      ; IS D0 ZERO ?
BNE ERR_Close ; IF NEGATIVE,
BRANCH TO
; Close WITH ERROR CODE
```

If there is no error, the second line is ignored. But if there is an error, the second line will branch to the short routine to leave the machine code with an error message

```
ERR_Close MOVE.W #SCA,A2 ;
UT_ERR0 IN A2
JSR (A2)
```

This causes an error message to print in channel 0, depending on the number in D0.

If you have some other code (not a Qdos call) which might get an error, you will have to devise your own test for the error (usually by comparing some data with result obtained), then if an error is detected, put a suitable negative number in D0 before branching to ERR_Close. This can be tricky, and there are no general rules for it. A simple example of this appears in one of the listings.

The other way of incorporating your machine code into a SuperBasic program is to make the machine code routine a SuperBasic procedure or function. This means that when you Lbytes and Call the routine (with no extra parameters) the routine has a name which is added to the list of SuperBasic words. The machine code needs to call a short routine to add the procedure or function name to the SuperBasic list, and also to point to the code to be used when the name is used in SuperBasic.

Listing 1

```
*****
; 'CALLCHANNELID'
; *****
; THIS ROUTINE WILL TAKE THE FIRST PARAMETER OF THE CALL, WHICH GOES INTO
; REGISTER D1, AND CONVERTS IT TO A BASIC CHANNEL ID IN A0.
; IT SHOULD BE THE FIRST PART OF THE CALL CODE.
; AS IT IS THE START OF A ROUTINE WHICH WILL CORRUPT REGISTERS, THE FIRST
; LINE IS TO STORE REGISTERS. THESE REGISTERS SHOULD BE RESTORED BEFORE
; RETURNING TO BASIC.
;
; START          MOVEM.L  A0-A7/D1-D7,-(A7) ; STORE REGISTERS
;                MOVEQ    #28,D0      ; #CH_LENCH IN D0
;                MULU     D1,D0        ; MULTIPLY CH. NO. BY ENTRY LENGTH
;                MOVE.L    $30(A6),A2  ; BV_CHBAS IN A2
;                ADDA.L    D0,A2        ; ADD TO GET CHANNEL ID ADDRESS
;                MOVE.L    0(A6,A2.L),A0 ; CHANNEL ID IN A0
;
; *****
```

Listing 2

```
*****
; 'DEFPROC'
; *****
; THIS ROUTINE WILL ADD A LIST OF PROCEDURES AND FUNCTIONS TO THE
; SUPERBASIC LIST
;
;                MOVE.W    $110,A2    ; BP_INIT IN A2
;                LEA       PROC_DEF,A1 ; BASIC PROCEDURE DEFINITIONS
;                JMP       (A2)
;
; NEXT IS THE PROCEDURE & FUNCTION DEFINITION TABLE. ADJUST TO SUIT.
; THE FIRST WORD SHOULD BE THE NUMBER OF PROCEDURES. BUT IF THE NAMES ARE
; LONGER THAN 7 BYTES, IT WILL NEED MORE.
;
; .PROC_DEF      DC.W      2          ; ONE LONG NAME PROCEDURE
; THIS IS FOLLOWED BY THE NEXT THREE LINES REPEATED FOR EACH PROCEDURE
;
;                DC.W      START-*    ; POINTER TO ROUTINE
;                DC.B      8          ; LENGTH OF NAME
;                DC.B      'ELLIPSES' ; PROCEDURE NAME
; THEN A 0 TO INDICATE THE END OF THE PROCEDURES
;
;                DC.W      0          ; END OF PROCEDURES
; THIS IS FOLLOWED BY EXACTLY THE SAME PATTERN FOR FUNCTIONS
;
;                DC.W      0          ; NO FUNCTIONS
;                DC.W      0          ; END OF DEFINITIONS
;
; *****
```

Listing 3

```
*****
; 'PROCCHANNELID'
; *****
; THIS ROUTINE WILL TAKE THE ONE PARAMETER OF A PROCEDURE AND TURN IT
; INTO A BASIC CHANNEL ID IN A0
; IT SHOULD COME AT THE START OF THE PROCEDURE, AFTER STORING REGISTERS.
;
;                MOVE.W    $118,A2    ; CA_GTLIN IN A2
;                JSR       (A2)
;                MOVEQ     #-15,D0    ; ERR_BP IN D0
;                CMPI.W    #1,D3      ; ONE PARAMETER ?
;                BNE       ERR_CLOSE ; IF NOT, CLOSE
;                MOVE.L    0(A6,A1.L),D0 ; PARAMETER IN D0
;                MOVEQ     #28,D1     ; CH_LENCH IN D1
;                MULU      D1,D0
;                MOVE.L    $30(A6),A2 ; BV_CHBAS IN A2
;                ADDA.L    D0,A2
;                MOVE.L    0(A6,A2.L),A0 ; CHANNEL ID IN A0
;
; *****
```

Listing 4

```
*****
; 'ERR_CLOSE'
; *****
; WHEN AN ERROR HAS BEEN DETECTED, AND THE ERROR CODE IS IN D0, A BRANCH
; TO THIS CODE WILL PRINT AN ERROR MESSAGE IN BASIC CHANNEL #0
; THIS CODE SHOULD BE FOLLOWED BY WHATEVER CODE IS NEEDED TO MAKE A SAFE
; RETURN TO BASIC.
;
; .ERR_CLOSE     MOVE.W    $CA,A2     ; UT_ERR0 IN A2
;                JSR       (A2)
;
; *****
```

Listing 5

```
*****
; 'ELLIPSES CALL CODE'
; *****
; 'CALLCHANNELID'
;
; .START          MOVEM.L  A0-A7/D1-D7,-(A7) ; STORE REGISTERS
;                MOVEQ    #28,D0      ; #CH_LENCH IN D0
;                MULU     D1,D0        ; MULTIPLY CH. NO. BY ENTRY LENGTH
;                MOVE.L    $30(A6),A2  ; BV_CHBAS IN A2
```

Adding routines

More about this later. This is what Toolkits usually do, and Simon Goodwin's excellent *DIY Toolkit* series shows lots of examples of this approach to adding machine code routines to SuperBasic. I have no wish to duplicate his efforts, and I suggest you refer to his articles to discover more. I would simply add the point that you do not have to stick to Toolkit routines, that is, ones that might have a variety of uses in different programs. You can use the same technique for machine code written specifically to go into one particular program for one particular purpose.

To illustrate how all this works, I am going to use the Ellipse program put together in part 8 of this series. I think it will be informative to see the same basic routine as both a free standing program, and as part of a larger program. There are a number of modifications that need to be made, both to make it work as a subroutine, and to make it more flexible.

1. Since it is not to be a free standing program, it does not need Jobstart or Endjob, so these are removed. However, the routine will need to end with the two lines mentioned earlier, so that it returns to SuperBasic successfully.

2. As part of a SuperBasic program, it would be better to use SuperBasic console channels, and enter the channel number as a CALL parameter (or as a procedure parameter, if you have done it that way). This means it does not need Console and Close to open and close a console window, so these are removed. There is also no need to activate a cursor in the SuperBasic console windows, so we remove Cursor.

3. As we are not going to close the console window, we need to restore it to a normal print mode from the XOR mode used in the Ellipse routine, so we need to add a Writemode at the end of the routine to restore the normal print ink colour onto paper colour mode.

Error trapping

4. We need to include some appropriate error trapping. In this case, the main source of any error will be the channel number entered as a parameter. The number used has to be that of a channel which has been opened, obviously. Also, the Fbyte subroutine requires this to be a console channel. In the case of the code being used as a SuperBasic procedure, we must also check that only one parameter is used. (Actually, we could ignore the extra parameters if more than one is entered, but it is good form to check anyway.)

5. We need some code to turn our parameter into the channel ID needed in register A0. Also, in the case of the SuperBasic procedure, we need some code to get the parameter from the SuperBasic in the first place. Since these

two points are the biggest innovations in this article, we shall look at them in some detail.

The first important point to remember is that just about everything the QL stores about its SuperBasic is in an address which is stored relative to the address in register A6. This is done so that as the demand for ram space changes, Qdos can simply move the whole lot to a new set of locations, but only has to change the address in A6 to enable us to work out the new position of everything else. The SuperBasic channel IDs are stored in addresses starting from \$30 above A6. So we can get the address into register A2 with the command:

```
MOVE.L #$30(A6),A2
```

Each channel entry is \$28 bytes long, and they run in sequence from channel #0 upwards. So if we multiply our channel number by \$28, and add that to the address we have put in A2, we get the address of the channel entry for the channel we want (relative to A6, of course!) The long word at this address is the channel ID. So we can put the channel ID in register A0, where it is needed for our routines, using the line

```
MOVE.L 0(A6,A2.L),A0
```

```

      ADDA.L    DO,A2      ; ADD TO GET CHANNEL ID ADDRESS
      MOVE.L    0(A6,A2.L),A0 ; CHANNEL ID IN A0
;
; *****
; THE REST OF THIS IS COMMON TO BOTH THIS PROGRAM AND THE CODE NEEDED
; TO MAKE 'ELLIPSES' A SUPERBASIC PROCEDURE
;
.WMODE      MOVEQ    #$2C,DO ; #SETMD IN DO
             MOVE.W    #$FFFF,D1 ; XOR INK INTO BACKGROUND
             TRAP      #3      ;
;
.COPY        LEA.L    DEFAULTS,A1 ; DEFAULT PARAMS ADDRESS IN A1
             LEA.L    EXA,A2      ; ACTUAL PARAMS ADDRESS IN A2
             MOVEQ    #4,D1      ; LOOP COUNTER
.LOOP        MOVE.L    (A1)+,(A2)+ ; MOVE TWO PARAMETERS
             DBRA      D1,LOOP    ; DECREMENT AND LOOP
             BRA.S     FIRST      ; DRAW DEFAULT ELLIPSE
;
.MAIN        JSR      FBYTE      ; GET KEY PRESS
; AN INVALID CHANNEL ID WILL GENERATE AN ERROR HERE TO BE TRAPPED
; ---- ADD NEXT TWO LINES TO TRAP ERRORS ----
             TST.L     DO        ; ERROR ?
             BNE      ERR_CLOSE
             LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
             CMPI.B    #10,(A1)   ; IS BYTE = 10 <ENTER>?
             BEQ       CLOSE     ; IF SO, CLOSE
             LEA.L     BYTE,A1    ; BYTE ADDRESS IN A1
             CMPI.B    #32,(A1)   ; IS BYTE = 32 <SPACE>
             BEQ.S     COPY       ; IF SO, NEW ANOTHER ELLIPSE
             JSR      ELLIPSE    ; TO REMOVE OLD ELLIPSE
             JSR      TEST       ; TEST FOR KEY PRESS
.FIRST       JSR      ELLIPSE    ; TO DRAW NEW ELLIPSE
; AN INVALID CHANNEL ID WILL GENERATE AN ERROR HERE TO BE TRAPPED
; ---- ADD NEXT TWO LINES TO TRAPP ERROR ----
             TST.L     DO        ; ERROR?
             BNE      ERR_CLOSE
             BRA       MAIN      ; AND START AGAIN
;
.FBYTE       LEA.L     BYTE,A1    ; BYTE STORE ADDRESS IN A1
             MOVEQ    #1,DO      ; IO_FBYTE IN DO
             MOVE.W    #0,D1      ; CLEAR WORD IN D1
             MOVE.W    #$FFFF,D3 ; INFINITE TIMEOUT
             TRAP      #3
             MOVE.B    D1,(A1)   ; STORE BYTE
             RTS
;
.BYTE        DC.B      0
; ---- DELETE LINE TO SET ADDRESS OF RI STACK ----
.ELLIPSE
;
; MAKE ROOM FOR 1ST INTEGER
             SUBQ.L    #2,A1
; PUT FIRST INTEGER ON RI STACK
;
; CONVERT TO F.P.
             MOVE.W    EXA,0(A6,A1.L) ;
             MOVEQ    #8,DO      ; #RI_FLOAT ON DO
             MOVE.W    #11C,A3   ; RI_EXEC ON A3
             JSR      (A3)
; REPEAT FOR 2ND INTEGER
             SUBQ.L    #2,A1
             MOVE.W    EXB,0(A6,A1.L) ;
             MOVEQ    #8,DO
             JSR      (A3)
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE X COORD ON RI STACK.
             MOVEQ    #10,DO     ; #RI_DIV ON DO
             JSR      (A3)
;
; NOW REPEAT THE WHOLE THING FOR THE Y COORDINATE
; MAKE ROOM FOR 1ST INTEGER
             SUBQ.L    #2,A1
; PUT FIRST INTEGER ON RI STACK
             MOVE.W    EYA,0(A6,A1.L) ;
; CONVERT TO F.P.

```

If this all sounds too much to handle, don't worry. **Listing two** is a short routine to do this for you. If you Call your routine from SuperBasic with the channel number as the one parameter after the address, this routine at the start of your code will put the channel ID in A0 for you. The only part of this not already mentioned is the first line. Since your routine will alter the registers, they must first be saved on the stack (except D0, of course). Clearly, at the end of your routine, you must restore the registers before returning to SuperBasic.

To SuperBasic

Listing 2 is how you add your routine to the SuperBasic list of procedures and functions. It follows a very simple format, which should cause no problems, once you have tried it a few times. It starts with the routine itself, which is to load the address of the vectored routine BP_INIT into A2. The address of the definitions table is loaded into A1, and then there is a jump to the vectored routine.

The definition table itself follows a fixed format, as follows:

We start with a word for the number of procedures to be added. This word allocates space in the procedures table, but if the average name length is more than seven, then more space is needed. With one eight-character name, I have made this word equal to 2. Then for each procedure to be added, this word is followed by a word for the address of the routine, a byte for the length of the procedure name, and a byte string for the name itself. A zero word then indicates the end of the procedures. Exactly the same pattern follows for the functions. In our case, we have no functions to add, so we have a zero word for the number of functions and a zero word for the end of the functions. This pattern is very simple to modify to add any number of procedures and functions to SuperBasic.

Channel ID

Listing three is the routine to convert the SuperBasic channel number, passed with the procedure, into a channel ID in A0. It starts with a jump to the vectored routine CA_GTLIN, which will fetch an indeterminate number of long word arguments passed when a procedure is called from SuperBasic. These long words are stacked up on the arithmetic stack, and the number of long words passed is returned as the word in D3.

At this point we put in our DIY error trapping for the number of parameters. We put -15 in D0, and if it is still there when we return from the procedure, it will generate a bad parameter message in channel 0, and cause the SuperBasic program to stop. We then compare D3 with 1, and if it is not 1 we branch to the

```

        MOVEQ    #8,D0      ; #RI_FLOAT ON D0
        JSR      (A3)      ;
; REPEAT FOR 2ND INTEGER
        SUBQ.L   #2,A1      ;
        MOVE.W   EYB,0(A6,A1.L) ;
        MOVEQ    #8,D0      ;
        JSR      (A3)      ;
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE Y COORD ON RI STACK.
        MOVEQ    #10,D0     ; #RI_DIV ON D0
        JSR      (A3)      ;
;
; NOW REPEAT THE WHOLE THING FOR THE ECCENTRICITY
; MAKE ROOM FOR 1ST INTEGER
        SUBQ.L   #2,A1      ;
; PUT FIRST INTEGER ON RI STACK
        MOVE.W   ECCA,0(A6,A1.L) ;
; CONVERT TO F.P.
        MOVEQ    #8,D0      ; #RI_FLOAT ON D0
        JSR      (A3)      ;
; REPEAT FOR 2ND INTEGER
        SUBQ.L   #2,A1      ;
        MOVE.W   ECCB,0(A6,A1.L) ;
        MOVEQ    #8,D0      ;
        JSR      (A3)      ;
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE ECCEN. ON RI STACK.
        MOVEQ    #10,D0     ; #RI_DIV ON D0
        JSR      (A3)      ;
;
; NOW REPEAT THE WHOLE THING FOR THE RADIUS
; MAKE ROOM FOR 1ST INTEGER
        SUBQ.L   #2,A1      ;
; PUT FIRST INTEGER ON RI STACK
        MOVE.W   RADA,0(A6,A1.L) ;
; CONVERT TO F.P.
        MOVEQ    #8,D0      ; #RI_FLOAT ON D0
        JSR      (A3)      ;
; REPEAT FOR 2ND INTEGER
        SUBQ.L   #2,A1      ;
        MOVE.W   RADB,0(A6,A1.L) ;
        MOVEQ    #8,D0      ;
        JSR      (A3)      ;
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE RADIUS ON RI STACK.
        MOVEQ    #10,D0     ; #RI_DIV ON D0
        JSR      (A3)      ;
;
; NOW REPEAT THE WHOLE THING FOR THE ANGLE OF ROTATION
; MAKE ROOM FOR 1ST INTEGER
        SUBQ.L   #2,A1      ;
; PUT FIRST INTEGER ON RI STACK
        MOVE.W   ROTA,0(A6,A1.L) ;
; CONVERT TO F.P.
        MOVEQ    #8,D0      ; #RI_FLOAT ON D0
        JSR      (A3)      ;
; REPEAT FOR 2ND INTEGER
        SUBQ.L   #2,A1      ;
        MOVE.W   ROTB,0(A6,A1.L) ;
        MOVEQ    #8,D0      ;
        JSR      (A3)      ;
; NOW DIVIDE ONE F.P. NUMBER BY THE OTHER TO LEAVE ROTATION ON RI STACK.
        MOVEQ    #10,D0     ; #RI_DIV ON D0
        JSR      (A3)      ;
;
; NOW PLOT THE ELLIPSE
        MOVEQ    #33,D0     ; #SD.ELIPS ON D0
        ADDA.L   A6,A1      ; MAKE A1 STACK ABSOLUTE
        TRAP     #3         ;
        RTS          ;
;
; INTEGER TABLE
.EXA      DC.W    80        ; 1ST INTEGER OF CENTRE X
.EXB      DC.W    1         ; 2ND INTEGER OF CENTRE X
.EYA      DC.W    50        ; 1ST INTEGER OF CENTRE Y
.EYB      DC.W    1         ; 2ND INTEGER OF CENTRE Y
.ECCA     DC.W    10        ; 1ST INTEGER OF ECCENTRICITY
.ECCB     DC.W    10        ; 2ND INTEGER OF ECCENTRICITY
.RADA     DC.W    20        ; 1ST INTEGER OF RADIUS
.RADB     DC.W    1         ; 2ND INTEGER OF RADIUS
.ROTA     DC.W    31        ; 1ST INTEGER OF ROTATION
.ROTB     DC.W    10        ; 2ND INTEGER OF ROTATION
;
; DEFAULTS
        DC.W    80        ; 1ST INTEGER OF CENTRE X
        DC.W    1         ; 2ND INTEGER OF CENTRE X
        DC.W    50        ; 1ST INTEGER OF CENTRE Y
        DC.W    1         ; 2ND INTEGER OF CENTRE Y
        DC.W    10        ; 1ST INTEGER OF ECCENTRICITY
        DC.W    10        ; 2ND INTEGER OF ECCENTRICITY
        DC.W    20        ; 1ST INTEGER OF RADIUS
        DC.W    1         ; 2ND INTEGER OF RADIUS
        DC.W    31        ; 1ST INTEGER OF ROTATION
        DC.W    10        ; 2ND INTEGER OF ROTATION
;
; TEST
        LEA.L    BYTE,A1    ; BYTE ADDRESS IN A1
        CMPI.B   #210,(A1)  ; IS BYTE = <CTRL><UP> ?
        BEQ.S    UPFP       ; IF 80, THEN UP
        LEA.L    BYTE,A1    ; BYTE ADDRESS IN A1
        CMPI.B   #218,(A1)  ; IS BYTE = <CTRL><DOWN> ?
    
```


ERR_Close routine, which means the program will stop with an error message. If there is no error, we move the parameter to D0. Remember that A1 points to the stack, relative to A6, so to do this we need the line

```
MOVE.L 0(A6,A1.L),D0
```

The rest of this is to turn the parameter into a channel ID, and is the same as **Listing one**.

Listing four is very short and simple. It is the routine which is branched to when an error is detected, and if there is an error code in D0, it will print the error message in channel 0. This code should be followed by whatever is needed to make a safe return to SuperBasic. This normally means returning registers to their previous state, and possibly also making sure the write mode for any windows is back to normal.

Application

We are now going to apply all this to making our Ellipses program into a routine to use in a SuperBasic program, first as a piece of Call code, and then as a SuperBasic procedure. In both cases, the major part of the listing will be the same. Only the front end of the code will be different in the two cases. In order not to repeat large amounts of listing, I have put in the complete listing for the Call routine as **Listing five**, and the modification needed to make Ellipse a SuperBasic procedure as **Listing six**. If you have been following this series and have already typed in the program in part 8, you can use that as a starter for **Listing five**, and make the appropriate alterations. The following assumes that is what you will be doing.

The common code starts with .WMODE, and anything before that can be deleted. In the .MAIN routine, Fbyte will generate an error if the channel is not open, or if it is not a con_channel, so this error must be trapped. Similarly, Ellipse may generate an error to be trapped. Right at the end of the code we must add the ERR_Close routine to print any error message for trapped errors, and .Close will restore the window used to normal print mode, and restore the registers. At the front of all this is merged Listing 1, which will store the registers, and turn the parameter of the Call into a channel ID. And that's it.

To make Ellipse a SuperBasic procedure, we simply merge **Listing six** at the front instead of **Listing one**. This is essentially a combination of **Listings two** and **three** to add Ellipses to the procedure list, and to turn the procedure argument into a channel ID. One extra line is added to store the registers. This is to balance the restore registers line at the end of **Listing five**. As I am not too familiar with the inner workings of SuperBasic, I am not sure this storing and restoring registers is actually needed here, because I suspect that SuperBasic deals with this

```

      BEQ.S      DOWNFP      ; IF SO THEN DOWN
      BRA       UPDOWN      ; IF NEITHER, SKIP TO NEXT BIT

;
; .UPFP
      LEA.L      RADA,A1     ; RADA ADDRESS IN A1
      LEA.L      RADB,A2     ; RADB ADDRESS IN A2
      MOVE.W      (A1),D0     ; RADA IN D0
      CMPI.W      #100,D0     ; RADA ALREADY 100 ?
      BEQ        NEXTBIT     ; IF SO LEAVE UNCHANGED
      ADDI.W      #1,(A1)     ; ADD 1 TO RADA
      MOVE.W      (A1),D0     ; RADA TO D0
      CMPI.W      #100,D0     ; IS RADA =100 ?

```

Listing 6

```

; *****
; 'ELLIPSES PROCEDURE CODE'
; *****
; THIS CODE MUST BE PLACED IN FRONT OF THE COMMON CODE FROM LISTING 5
; TO MAKE THE NAME 'ELLIPSES' INTO A SUPERBASIC PROCEDURE
;
      MOVE.W      $110,A2     ; BP_INIT IN A2
      LEA         PROC_DEF,A1 ; BASIC PROCEDURE DEFINITIONS
      JMP         (A2)        ;
;
; .PROC_DEF
      DC.W        2           ; ONE LONG NAME PROCEDURE
      DC.W        START-*    ; POINTER TO ROUTINE
      DC.B        8           ; LENGTH OF NAME
      DC.B        'ELLIPSES' ; PROCEDURE NAME
      DC.W        0           ; END OF PROCEDURES
      DC.W        0           ; NO FUNCTIONS
      DC.W        0           ; END OF DEFINITIONS
;
; .START
; ---- ADD NEXT LINE TO STORE REGISTERS ----
      MOVEM.L     D1-D7/A0-A7,-(A7) ; STORE REGISTERS
      MOVE.W      $118,A2     ; CA_GTLIN IN A2
      JSR         (A2)
      MOVEQ       #-15,D0     ; ERR_BP IN D0
      CMPI.W      #1,D3       ; ONE PARAMETER ?
      BNE        ERR_CLOSE    ; IF NOT, CLOSE
      MOVE.L      0(A6,A1.L),D0 ; PARAMETER IN D0
      MOVEQ       #$28,D1     ; CH_LENCH IN D1
      MULU        D1,D0
      MOVE.L      $30(A6),A2 ; BV_CHBAS IN A2
      ADDA.L      D0,A2
      MOVE.L      0(A6,A2.L),A0 ; CHANNEL ID IN A0
;
; *****
;
; .UP3
      LEA.L      ECCA,A1     ; INTEGER ADDRESS IN A1
      ADDI.W      #1,(A1)     ; ADD 1 TO INTEGER
      BRA        NEXTBIT     ; SKIP DOWN
;
; .DOWN3
      LEA.L      ECCA,A1     ; INTEGER ADDRESS IN A1
      SUBI.W      #1,(A1)     ; SUBTRACT 1 FROM INTEGER
      BRA        NEXTBIT     ;
;
; .UPDOWN4
      LEA.L      BYTE,A1     ; BYTE ADDRESS IN A1
      CMPI.B      #211,(A1)   ; IS BYTE = <ALT><CTRL><UP> ?
      BEQ.S      UP4         ; IF SO, THEN UP
      LEA.L      BYTE,A1     ; BYTE ADDRESS IN A1
      CMPI.B      #219,(A1)   ; IS BYTE = <ALT><CTRL><DOWN> ?
      BEQ.S      DOWN4       ; IF SO, THEN DOWN
      BRA        NEXTBIT     ; IF NEITHER, SKIP TO NEXT BIT
;
; .UP4
      LEA.L      ROTA,A1     ; INTEGER ADDRESS IN A1
      ADDI.W      #1,(A1)     ; ADD 1 TO INTEGER
      BRA        NEXTBIT     ; SKIP DOWN
;
; .DOWN4
      LEA.L      ROTA,A1     ; INTEGER ADDRESS IN A1
      SUBI.W      #1,(A1)     ; SUBTRACT 1 FROM INTEGER
;
; .NEXTBIT
      RTS
;
; ---- ADD THE FOLLOWING LINES TO CLOSE ROUTINE AND RETURN TO BASIC ----
; .ERR_CLOSE
      MOVE.W      $CA,A2     ; UT_ERRO IN A2
      JSR         (A2)
;
; .CLOSE
      MOVEQ       #$2C,D0     ; #SETMD IN D0
      MOVE.W      #0,D1      ; NORMAL PRINT MODE
      TRAP        #3
      MOVEM.L     (A7)+,D1-D7/A0-A7 ; RESTORE REGISTERS
      RTS
;
; *****
;
      BNE        NEXTBIT     ; IF NOT, NEXT BIT
      MOVE.W      (A2),D0     ; RADB TO D0
      CMPI.W      #1,D0       ; IS RADB = 1 ?
      BEQ        NEXTBIT     ; THEN LEAVE RADA,RADB UNCHANGED
      MOVE.W      #10,(A1)    ; ELSE, RADA = 10
      MOVE.W      (A2),D0     ; RADB IN D0
      MOVE.W      #10,D1      ; 10 IN D1
      DIVU        D1,D0       ; RADB/10
      MOVE.W      D0,(A2)     ; NEW DO VALUE IN RADB
      BRA        NEXTBIT     ; SKIP TO NEXT BIT
;
; MAKE NUMBER SMALLER.
; .DOWNFP
      LEA.L      RADA,A1     ; RADA ADDRESS IN A1
      LEA.L      RADB,A2     ; RADB ADDRESS IN A2
      MOVE.W      (A1),D0     ; RADA TO D0
      CMPI.W      #9,D0       ; RADA ALREADY 9 ?
      BEQ        NEXTBIT     ; IF SO, LEAVE UNCHANGED
      SUBI.W      #1,(A1)     ; SUBTRACT 1 FROM RADA

```

already. However, it certainly won't do any harm, and anyway, it's a good policy to make sure that whenever you do anything which may corrupt the registers, you always store them beforehand and then restore them afterwards.

Test program

Listing seven is a short SuperBasic program to test the working of both the Call code and the Ellipses procedure. All it does is to open a random set of channels numbering 3 to 9, to add to the SuperBasic default channels 0, 1 and 2. It then gives you the choice of using either the Call code or the Ellipses procedure, followed by the choice of which channel to draw in. You can then draw circles and ellipses all over the screen to your hearts content, noticing that both methods work equally well. The SuperBasic itself is straightforward and needs no explanation to those of you familiar with the language. The only point that needs making is that as it stands the program expects to find the Call code as flp1_ellcall and the procedure code as flp1_ellproc. You may need to alter this to suit your device and file names. Also, I have made no attempt to Respr exactly the right amount of ram space - 1000 bytes is more than enough.

Although we have only considered using machine code within SuperBasic programs, clearly, there is scope for using it with other languages. Often, languages like Forth and C are not designed for a particular computer, and are therefore unable to take advantage of many of the special facilities a machine like the QL can offer. This can be overcome by using machine code routines. The documentation with your language should tell you how to do it. However, a word of caution is needed. One of the big advantages of these languages is that the programs are portable, that is, a program written on one computer will run on a different computer. If you put QL specific machine code routines in, this will no longer be true. Your program will only run on a QL or compatible machine.

Conclusion

This is the end of the series for the present. Clearly my little Toolkit is not a complete list of all the routines that might be useful. For example, I have not touched the wide area of input and output to files in data bases, for the simple reason that the programming problems this throws up do not particularly interest me. But I hope it has managed to persuade some QL users, who at present only copy listings, that writing machine code is not as formidable as it at first seems. If you have followed this series through, you should be able to put some simple ideas into practice; in which case, please do have a go. Remember, new software is the lifeblood of a machine like the QL. Even if you can't produce something that can be sold commercially (let's face it - most of us can't), you can always see if *QL World* will print it, or perhaps Quanta or one of the PD libraries would like it. Happy coding!

```

MOVE.W (A1),D0 ; RADA TO DO
CMPI.W #9,D0 ; IS RADA = 9 ?
BNE NEXTBIT ; IF NOT, NEXT BIT
MOVE.W (A2),D0 ; RADB TO DO
CMPI.W #10000,D0 ; IS RADB = 10000 IF DIVIDING
BEQ NEXTBIT ; THEN LEAVE RADA,RADB UNCHANGED
MOVE.W #99,(A1) ; ELSE, RADA = 99
MOVE.W (A2),D0 ; RADB IN DO
MOVE.W #10,D1 ; 10 IN D1
MULU D1,D0 ; RADB*10
MOVE.W D0,(A2) ; NEW DO VALUE INTO RADB
BRA NEXTBIT ;
.UPDOWN LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #208,(A1) ; IS BYTE = 208 ?
BEQ.S UP ; IF SO, THEN UP
LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #216,(A1) ; IS BYTE = 216 ?
BEQ.S DOWN ; IF SO THEN DOWN
BRA UPDOWN2 ; IF NEITHER, SKIP TO UPDOWN2
.UP LEA.L EYA,A1 ; INTEGER ADDRESS IN A1
ADDI.W #1,(A1) ; ADD 1 TO INTEGER
BRA NEXTBIT ; SKIP DOWN
.DOWN LEA.L EYA,A1 ; INTEGER ADDRESS IN A1
SUBI.W #1,(A1) ; SUBTRACT 1 FROM INTEGER
BRA NEXTBIT ;
.UPDOWN2 LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #200,(A1) ; IS BYTE = <RIGHT>?
BEQ.S UP2 ; IF SO, THEN UP
LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #192,(A1) ; IS BYTE = <LEFT> ?
BEQ.S DOWN2 ; IF SO THEN DOWN
BRA.S UPDOWN3 ; IF NEITHER, SKIP TO UPDOWN3
.UP2 LEA.L EXA,A1 ; INTEGER ADDRESS IN A1
ADDI.W #1,(A1) ; ADD 1 TO INTEGER
BRA NEXTBIT ; SKIP DOWN
.DOWN2 LEA.L EXA,A1 ; INTEGER ADDRESS IN A1
SUBI.W #1,(A1) ; SUBTRACT 1 FROM INTEGER
BRA NEXTBIT ;
.UPDOWN3 LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #209,(A1) ; IS BYTE = <ALT><UP> ?
BEQ.S UP3 ; IF SO, THEN UP
LEA.L BYTE,A1 ; BYTE ADDRESS IN A1
CMPI.B #217,(A1) ; IS BYTE = <ALT><DOWN> ?
BEQ.S DOWN3 ; IF SO THEN DOWN
BRA.S UPDOWN4 ; IF NEITHER, SKIP TO UPDOWN4

```

Listing 7

```

100 init
110 REPEAT loop
120 CLS#0
130 PRINT#0,"SUPERBASIC PROCEDURE(P), CALL CODE(C) or QUIT(Q) ?"
140 a$=INKEY$(-1)
150 SELECT ON a$
160 = "p" : proc_code
170 = "c" : call_code
180 = "q"
190 FOR chan = 3 TO 9: CLOSE#chan
200 STOP
210 END SELECT
220 END REPEAT loop
230 STOP
240 DEFINE PROCEDURE init
250 MODE 0
260 OPEN#3,con_512x256a0x0:CLS#3
270 p = RESPR(1000)
280 c = RESPR(1000)
290 LBYTES flp1_ellproc,p
300 CALL p
310 LBYTES flp1_ellcall,c
320 FOR chan=3 TO 9
330 x = RND(100 TO 512)
340 y = RND(100 TO 200)
350 xo = RND(0 TO 512-x)
360 yo = RND(0 TO 200-y)
370 OPEN#chan,"con_512x256a0x0"&x"&y"&a"&x0"&x"&y0
380 INK#chan,RND(2 TO 7):BORDER#chan,2,RND(2 TO 7):OVER#chan,-1
390 END FOR chan
400 END DEFINE
410 DEFINE PROCEDURE proc_code
420 CLS#0:select_chan:prompts:ELLIPSES#ch
430 END DEFINE
440 DEFINE PROCEDURE call_code
450 CLS#0:select_chan:prompts:CALL c,ch
460 END DEFINE
470 DEFINE PROCEDURE select_chan
480 CLS#0
490 PRINT#0,"WHICH CHANNEL (0 to 9) ? "
500 REPEAT keypress
510 a$ = INKEY$(-1)
520 SELECT ON a$
530 = 0 TO 9
540 ch = a$: EXIT keypress
550 END SELECT
560 END REPEAT keypress
570 END DEFINE
580 DEFINE PROCEDURE prompts
590 CLS#0
600 PRINT#0,"CURSOR KEYS TO MOVE ELLIPSE AROUND"
610 PRINT#0,"CTRL+UP/DOWN TO CHANGE SIZE, ALT+UP/DOWN TO CHANGE ECCENTRICITY"
620 PRINT#0,"ALT+CTRL+UP/DOWN TO ROTATE ELLIPSE"
630 PRINT#0,"SPACE TO DRAW ANOTHER ELLIPSE IN THE SAME WINDOW"
640 PRINT#0,"ENTER TO RETURN FROM ROUTINE"
650 END DEFINE

```

DJC

Dilwyn Jones Computing

41 Bro Emrys, Tal-y-Bont,

Bangor, Gwynedd LL57 3YT U.K.

Tel: Bangor (0248) 354023

NEW QL SOFTWARE

THE MEGA TOOL KIT £25.00
version on EPROM ..£40.00

(F 1M 128k) A superb new toolkit of over 200 SuperBASIC extensions. Ideal for use in BASIC programs and in compiled BASIC programs too—there is even a runtime version which can be used in compiled commercial or public domain programs free of royalty payment! Comprehensive manual and large file of demonstration routines! If you buy the EPROM version, you get the disk version too!

BANTER£25.00

(F 512k) A program for making large text banners, up to four pages long! Prints sideways, using outline fonts for greatly improved quality of printing (banish those horrible jagged edges!). Ideal program for making signs for shows, small businesses, adverts, and so on. Currently only suitable for 9 pin Epson compatible printers, other versions being written, please enquire!

S_EDIT£20.00

(F 384k) A really simple to use menu driven editor. Can be used for editing plain text files, binary files, creating source files for assemblers and so on—even to edit SuperBASIC programs, in fact any application where an easy to use editor is more useful than a word processor. Features include full block handling, search and replace, word wrap, margins, can use Toolkit 2 default device, dynamic linked list memory handling, configurable with CONFIG, many features!

OPD TRANSFER£15.00

Software to allow the transfer of files between the QL and the microdrives on OPD and Tonto computers, written by the author of Discover and Textidy.

NEW! PC VERSION OF
FLEET TACTICAL

COMMAND II£69.95

QL+PC versions pack £85.00

Upgrade—contact us!

Compatible with the QL version, but with improved graphics. Can be networked to a QL version by means of a QL to PC serial lead (wiring diagram included). If you enjoyed playing the QL version and also have a PC, you can now enjoy playing this on a PC as well!

PROGRAMMING

EASYPTR II£49.00

Budget EasyPtr£35.00

DISA disassembler£29.00

BASIC REPORTER ...£10.00

QLIBERATOR£50.00

BUDGET VERSION...£25.00

QLOAD & QREF£15.00

GRAPHICS

THE PAINTER£25.00

THE CLIPART£12.00

QRACTAL£20.00

QRACTAL SCREENS .£5.00

IMAGE PROCESSOR£15.00

Version 1 upgrade£10.00

PD2 CLIPART£10.00

SCREEN SNATCHER£10.00

TEXT 'N'GRAPHIX£20.00

TRANS24£10.00

DTP

PAGE DESIGNER 2+ £40.00

NB Delayed—please check if available yet before ordering.

UPGRADE PD2 TO PD2 PLUS£20.00

TEXT

BIBLE TEXT DISKS, PLAIN

TEXT FORMAT£20.00

QUILL_DOC FILES ...£20.00

SPELLBOUND£30.00

SPELLBOUND S.E....£50.00

Upgrade to SE version £30.00

QUICK POSTERS£10.00

ROB ROY PACK£10.00

DATABASE

ADDRESS BOOK & LABEL

PRINTER£15.00

DATA DESIGN£50.00

FLASHBACK S.E.....£40.00

QL GENEALOGIST ...£12.00

DBEASY£15.00

DBPROGS£15.00

LEISURE SOFTWARE

SOLITAIRE£15.00

CRICKET SECRETARY £12.00

QUESTION MASTER £10.00

QL FLEET TACTICAL

COMMAND II£49.95

THE FUGITIVE£9.95

COCKTAILS WAITER £10.00

OTHER QL SOFTWARE

PRINTMASTER£20.00

HOME BUDGET£20.00

REMIND-ME£12.00

REMIND-ME PLUS ...£20.00

SCREEN ECONOMISER

.....£12.00

SLOWGOLD£5.00

TASKMASTER£25.00

DISK LABELLER£10.00

THE CAT£5.00

DISPLAY SOFTWARE

VISION MIXER 1£10.00

VISION MIXER PLUS £22.50

PICTUREMASTER£15.00

PICTUREMASTER + .£20.00

FILE HANDLING

FILES 2£12.00

FILEMASTER£12.00

THE GOPHER£12.00

WINBACK£25.00

FILE TRANSFER

DISCOVER£20.00

MULTI DISCOVER£30.00

TEXTIDY£15.00

HARDWARE

MINI PROCESS

CONTROLLER£59.95

MPC TOOLKIT software £9.95

NETWORK PROVER ..£3.50

THE SMALL PRINT! POSTAGE AND PACKING CHARGES: Software is sent post-free to UK addresses. To other countries, please add £1.00 per program for postage and packing. PRICES: All prices are shown in UK Pounds Sterling. PAYMENT: We can accept payment by cheque (in UK Pounds Sterling currency only, please, drawn on UK branch of a bank or building society), Eurocheque, Postal Order, or by these credit cards: VISA, ACCESS, MASTERCARD, EUROCARD and by CONNECT card. Please state the card type, number, expiry date, your address and sign orders sent by post. We can also accept orders paid by credit card by telephone. There is an answering machine when I cannot answer in person, so that I can call you back later. Goods remain the property of DJC until paid for in full. PLEASE STATE IF YOU REQUIRE 3.5 OR 5.25 INCH DISKS OR ON MICRODRIVE CARTRIDGE (ADD £1 PER PROGRAM ON MICRODRIVE CARTRIDGE TO COVER HIGHER COSTS OF SUPPLYING SOFTWARE ON MICRODRIVE).